UNIVERSITY OF MALAYA

SESSION 2003/2004

PEER-TO-PEER APPLICATION USING MICROSOFT .NET FRAMEWORK

WXES 3181 PROJEK ILMIAH TAHAP AKHIR I

AND

WXES 3182 PROJEK ILMIAH TAHAP AKHIR II

TAN BOON YIN

WEK 010270

SUPERVISOR: DR. ROSLI SALLEH

MODERATOR: MR. NOR BADRUL ANUAR

DATE OF SUBMISSION: 19 FEBRUARY 2004

Abstract

Peer-to-peer (P2P) applications such as Napster and Gnutella that communicate as peers sharing and receiving information are becoming commonplace as a means for users connected on large networks to take advantage of the vast resources available to them. The Microsoft .NET Framework provides a rich platform for building P2P applications.

The main objective of this thesis is to develop a simple, effective, attractive, interactive, convenient and user friendly instant messaging and file sharing P2P application. This thesis also explains the concepts that make up P2P applications. The P2P application model, discovering others peers, and querying peers for information are discussed.

Basically, this P2P application includes three modules, server module, client module and library module. Server module handles all the server functionalities like listening to connections and receiving messages. Client module handles all the client functionalities including call request and call initiation. Library module handles all the Windows Form Controls and also acts as a middle agent for the communication between client and server.

The Microsoft .NET is the software development platform to be used widely through out the project. This project also will emphasize the advantages of Microsoft .NET in developing the P2P application. Microsoft SQL Server 2000 or Microsoft SQL Server Desktop Engine (MSDE) will be used to manage the database while Windows XP Professional will be my project development platform.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Project Overview

1.2 Project Objective

1.3 Project Scope

1.4 Project Limitation

1.5 Expected Outcome

1.6 Project Schedule

1.7 Chapter Summary

1.1 Project Overview

Peer-to-Peer (P2P) is a kind of decentralize computing where computers communicate directly with one another. P2P allows computers, and their users, to tap unused resources – such as extra megahertz of processing power, gigabytes of storage – that would otherwise remain locked up in individual desktops.

P2P computing has been around for several decades. The internet conceived in 1969 was a P2P system. The goal of the original ARPANET was to share computing resources around the US. The challenge for this effort was to integrate different kinds existing networks as well as future technologies with one common network architecture that would allow every host to be an equal player. Early applications of P2P were Usenet and DNS. This put a tremendous work-load on the consumption of bandwidth, but also standards of addressing/communicating to other machines had to be established. Prior to Napster's launch in 1999, one of the earliest large-scale uses of P2P computing was in 1994, when two scientists at the Goddard space Flight Center in Maryland networked 16 processors together and created a single cluster computer.

The most important characteristic and feature of P2P is decentralization that leads to increased performance, scalability and disruptiveness. As a network grows bigger, all centralized points will become weak points in its infrastructure. Example: The internet and the DNS system, or transatlantic peering points. Opposed to the traditional client/server model of standalone ftp/http/mail/etc. servers, peers are constantly connected to the network and constantly exchange information.

P2P technology can be used in the different areas. The following table shows the applications of P2P technology:

Table 1.1: Different types of P2P applications

Types	Examples
File Sharing	Gnutella, Napster
Distributed Computing	Seti@Home, Distributed.net, DistributedScience
P2P Search Engine	OpenCOLA
P2P Communication	ICQ, IRC, Eggdrop, Aimster
Edge Services	Intel's upcoming edge services
Device Intercommunication	Jini, Bluetooth
Anonymity/Anti-Censorship	Freenet, Onion Routing, Hacktivismo, Red Rover

Communication is a key element when writing nearly any type of application. An application gains value when it becomes distributed and interacts with other resources available to it on the Internet or intranet. The most common model for communication over the Internet today is client/server, where there is a client that knows how to request information and post information to a server, and the server knows how to respond to requests from the client. A browser talking to a Web server is a common example of this model. Many browsers can send requests to the Web server, and the server does its job by listening for those requests and responding back to each of the browsers requesting or sending information (usually in the form of Web pages). In this model, the Web server cannot arbitrarily contact the browser. The "conversation" is always initiated by the client.

A P2P application is different from the traditional client/server model because the applications involved act as both clients and servers. That is to say, while they are able to request information from other servers, they also have the ability to act as a server and respond to requests for information from other clients at

the same time. This approach increases the amount of value that each node on the network can add because it not only takes information from a source, but it also has the ability to share that information with other sources.

Unlike a client/server model, where a client sends a request, the server returns the result after processing the request. P2P is a different model altogether since here the applications involved act as both clients and servers. They are able to act as clients as they can request for information from other servers and can act as servers too as they can respond to requests for information from other clients at the same time. This approach increases the amount of value that each node on the network can add because it not only takes information from a source, but it also has the ability to share that information with other sources. This reduces the load on servers and allows them to perform specialized services (such as mail-list generation, billing, etc.) more effectively. Thus it can be said that a P2P network is one in which each workstation has equivalent responsibilities. Designed well, P2P can improve reliability, security, and availability because a single point of failure would not take down the system. It lets you use idle resources — that you already own – to expand and fortify storage, improve processing, and potentially reduce bandwidth requirements.

A typical P2P application has the following key features that help define it:

Discovering other peers

The application must be able to find other applications that are willing to share their information. Historically, the application finds these peers by registering with a central server that maintains a list of all applications currently willing to share and giving that list to any new applications as they connect to the network. However,

there are other means available, such as network broadcasting or discovery algorithms.

Querying peers for content

Once these peers have been discovered, the application can ask them for the content that is desired by the application. Content requests often come from users, but it is highly conceivable that the P2P application is running on its own and performing its query as a result of some other network request that came to it rather than a specific request made by a user at that machine.

Sharing content with other peers

In the same way that the peer can ask others for content, it can also share content after it has been discovered.

P2P Instant Messaging

The first modern use of P2P began with the launch of the ICQ instant messaging service in 1996. Unlike traditional Internet chat services (such as Internet Relay Chat), ICQ did not rely on chat servers to host chat rooms or channels where users could exchange messages in real time. Instead, ICQ connected users directly, with no server involved, thus creating true P2P connections.

The popularity of ICQ (at its height, the ICQ network had 50 million users) inspired several important imitators and competitors. America Online incorporated instant messaging within its proprietary commercial network, and later offered a Web-based interface to their network through the AOL Instant Messenger

application. (AOL also purchased ICQ, in 1998.) Microsoft entered the instant messaging wars in 1999 with its MSN Messenger application, and upped the ante in 2001 with the new Windows Messenger program, included with its Windows XP operating system. Yahoo! also entered the fray with its Yahoo! Messenger application, and in 1998 a group of independent programmers developed Jabber, an instant messaging application that promises to break free from traditional proprietary messaging networks. Today more than 200 million users exchange instant messages using one or more of these proprietary systems, making instant messaging the single largest application of P2P technology.

P2P File Sharing

The next phase in P2P was the creation of applications that promoted P2P file sharing. In 1999, 18-year-old college student Shawn Fanning developed what has since become perhaps the most notorious P2P application: Napster. Fanning, frustrated by the difficult nature of finding digital music files on the Web, coded his own application to find and swap music files on individual PCs. Napster tapped into a huge pent-up demand and, by the end of 2000, had more than 50 million users swapping primarily MP3 music files. Since most of these files consisted of copyrighted material, however, Napster soon found itself on the receiving end of several bruising music-industry lawsuits. These lawsuits (and subsequent court rulings in favour of the record companies) effectively shut Napster down, forcing it to filter out all copyrighted files and change its business model to one of subscription delivery of licensed material.

The mistake Fanning made with Napster was to implement a system that was not completely P2P in nature. Napster used a central server to store and serves an

index of files that were available on the millions of peer computers that comprised the Napster network. The courts could effectively shut down the entire network by forcing the index server offline.

This mistake was not repeated by the many P2P file-sharing services that sprang up in the wake of the anti-Napster lawsuits. These services, such as Gnutella, Aimster, and Audiogalaxy, are true P2P networks, with no central index servers involved. Instead, these networks utilize P2P distributed search services to locate files on request; when a file is found, the two peers then connect to each other to transfer the file from one to the other.

The .NET Framework has a significant range of choices when it comes to the type of application that you can create. When writing P2P application, it is important to understand how it will be used. This will make it easier to decide which .NET application model to use. In the case of P2P, four powerful application models (or application types) are available. A brief overview of these models follows.

Web Services

Found in the System.Web.Services namespace, the Web Services technology provides an excellent way to handle registration, discovery, and content lookup for your P2P application. A Web Service allows you to quickly write a class that listens for incoming requests, processes them as they arrive, and sends back useful information in the form of objects that are easily understood by the peer application.

Windows Forms

Found in the System.WinForms namespace, Windows Forms is the .NET Framework solution for writing the type of rich Windows-based GUI applications that help to make the P2P experience much more exciting. Windows Forms is the ideal technology for writing the GUI for the peer that lets your users log in, request, and share content.

Web Forms

Found in the System.Web namespace, the Web Forms technology makes it very easy to return HTML content to a peer application. This can be useful if you want to spice up your P2P application with general content about the service or advertisements about using the service. When the P2P application starts up and registers with the Web Service, it can also call a Web Forms application to get the latest HTML content from the server.

Service Process

Found in the System.ServiceProcess namespace, a service process (also known as a Windows NT service) is useful in P2P scenarios as a long-lived discovery server. In most cases, a Web Service is better for fulfilling the role of the discovery service (mentioned previously). However, in cases where the discovery mechanism is not using the HTTP protocol, a service process listening for some other protocol might be the best way to go.

1.2 Project Objective

The purpose of this project is to first and foremost gainer and understanding of the P2P concept and then implements it in a real world situation. P2P application is an application which provides P2P service among the user in the world. It is found to be a potentially powerful online communication tool that gives many benefits to all the users.

Among the most desired objectives of this project are:

- To provide a simple, effective, attractive and interactive P2P application to attract users to go instant messaging and file sharing.
- To provide a user friendly interface to allow every process to be accomplished on a computer-based environment.
- To improve the convenient, accessibility and quality of interactions between the users.
- To be reliable. P2P application ensures the reliability of service and information with fewer mistakes as the transferring is carried out online.

1.3 Project Scope

The core features of P2P application:

- A P2P application should be able to locate other peers in the network.
- Once an application is able to locate other peers, it should be able to communicate with them using messages.
- Once the communication is established with other peers, the application should be able to receive and provide information, such as content.

The main user of this application system including all users that wishes to instant messaging and file sharing.

The three main modules:

- Client Module – handles all the client functionalities including call request and call initiation.
- Server Module - handles all the server functionalities like listening to connections and receiving messages.
- Library Module – handles all the Windows Form Controls, and also acts as a middle agent for the communication between client and server.

1.4 Project Limitation

Every system regardless of the manual or the fully automated system, there will always be limitations and weaknesses. The limitations of the P2P application system to be developed are:

- Portability

The P2P application system will be developed in the Microsoft Windows environment. Thus, P2P application cannot be ported to other platform such as Macintosh or Linux-based system.

- Language Supports

The English will be used as the sole language in this P2P application system. Due to the constraint of time, P2P application system in Bahasa Melayu version will be added in the future as an enhancement to the system when required.

1.5 Expected Outcome

- A P2P application system that consists three modules: client module, server module and library module.
- A P2P application system that allows users to instant messaging and file sharing.
- A standard application with a user friendly graphical user interface that will ease the user to instant messaging and file sharing.
- A system that will attract both computer literate and illiterate to use the P2P application.

1.6 Project Schedule

Time management is very important to all project developers. It is an important attribute in determining how deep a problem has been analyzed, how comprehensive should a solution be designed, how complete can a source code be implemented and how thoroughly can the program be tested.

A project schedule is planned at the system studies and planning phase to ensure that effort is distributed within the prescribed time frame to make the best use of resource.

The project schedule is shown in the Figure 1.1.

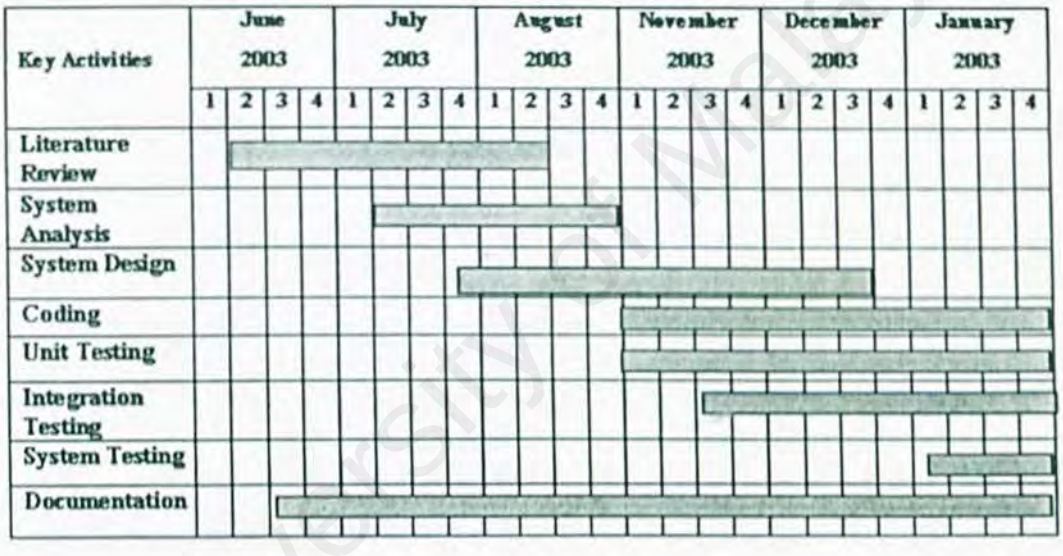


Figure 1.1: Project schedule

1.7 Chapter Summary

P2P application is the application of new technologies, particularly Internet and communication technologies to help group study, individuals businesses and other organizations conducting communication in a better way. Thus, P2P application is designed in built to fit in the trend of P2P technology.

With P2P application, users are able to communicate each other in a secure manner, instant messaging and file sharing in a more efficient manner.

Chapter 2 Literature Review

2.1 Purpose of Literature Review

A literature review is a background study about the knowledge and information gained to develop the project. The purpose is to:

- Enable us to do comparison on the past-developed projects.
- Study the strength and weakness of the existing developed projects and give an overview how to improve the weakness and fulfill the requirements needed for the project.
- Get a better understanding on the development tools that can be used to develop the project.
- Get a better knowledge on the development methodologies used while developing a project.

The following sections will take a closer look at the needs to develop the P2P application. These sections will include the following:

- Introduction to Peer-to-Peer [8][9][10]
- The Peer-to-Peer Existing System
- Microsoft .NET
- Development Platform
- Development Technology
- Database Management System
- Network

2.2 Introduction to Peer-to-Peer

A peer is a PC connected to the Internet, that can communicated directly with other PCs, using some communication protocol.

The Gartner Group defines P2P computing as: "characterized by direct connections using virtual namespaces, it describes a set of computing nodes that treat each other as equals (peers) and supply processing power, content or applications to other nodes in a distributed manner, with no presumptions about a hierarchy of control".

P2P computing based architecture allows for decentralized application design, moving from centralized server models to a distributed model where each peer, independent of software and hardware platforms, can benefit and profit from being connected to millions of other peers. In such architectures, clients and servers have a lateral relationship rather than the traditional vertical relationship, giving the whole peer group tremendous processing power and storage space.

It is worth mentioning that P2P computing is still in an evolving state and much needs to be done to overcome complex issues such as security, network bandwidth, and architecture designs.

Some of the core features of a P2P application:

- A P2P application should be able to locate other peers in the network.
- Once an application is able to locate other peers, it should be able to communicate with them using messages.
- Once the communication is established with other peers, the application should be able to receive and provide information, such as content.

Independently from the enterprise domain, a complementary type of distributed systems, the P2P networks, emerged. These huge networks follow a decentralized approach and harvest the resources of each peer. They do not depend on dedicated network servers to provide services. Each peer is considered as a server and a client simultaneously. A peer may request services from other peers while it provides services by itself to different peers. The P2P concept became popular with file sharing applications, like Napster, and Gnutella (Gnutella 2002).

In short, P2P networks may lead to a paradigm shift and may replace client/server architectures for several purposes.

2.2.1 Pure Peer-to-Peer

A pure P2P application does not involve any central server. It dynamically discovers other peers on the network and interacts with each of them for sending and receiving content. The advantage of this type of application is that it does not rely on any one server to be available for registration of its location in order for other peers to find it. However, the lack of a central discovery server poses a problem because a relatively low number of clients can be discovered, thereby limiting the application's reach. Also the time involved in discovering other peers is also more as the number of queries send over the network is more than the case in which you have a central server. In this scenario, a peer can either use information from a local configuration scheme to discover the clients (for example, a configuration entry that tells it who to talk with) or it can employ network broadcasting and discovery techniques such as IP multicast to discover the other peers.

Figure 2.1: A pure P2P

2.2.2 Peer-to-Peer with a Simple Discovery Server

This type of architecture is quite similar to the pure P2P model except that it relies on a central server for discovery of the other peers. However, this model involves registering with the central server also called as the discovery server at the time of log on. Thus the discovery server contains all the information of the peer who is logged on at a particular time. The peer application then uses this server to download list of the other peers participating on its network that it can use to query for content. When content is needed, it goes through the list and contacts each peer individually with its request. Thus the server only comes into picture for discovery of other peers and for obtaining a list of the peers who are logged on, and then it is up out application to contact all these peers individually.

However, this approach hinges on the availability of the central server. If the central server is not available, the P2P application will not be able to find other peers. In addition, requesting content from each individual peer can be quite expensive from a network resource perspective. This may not seem like a big deal if you're thinking about a few peers interacting over a network, but if your application is

being written for use over the Internet or a large enterprise environment, this consideration suddenly becomes much more significant.

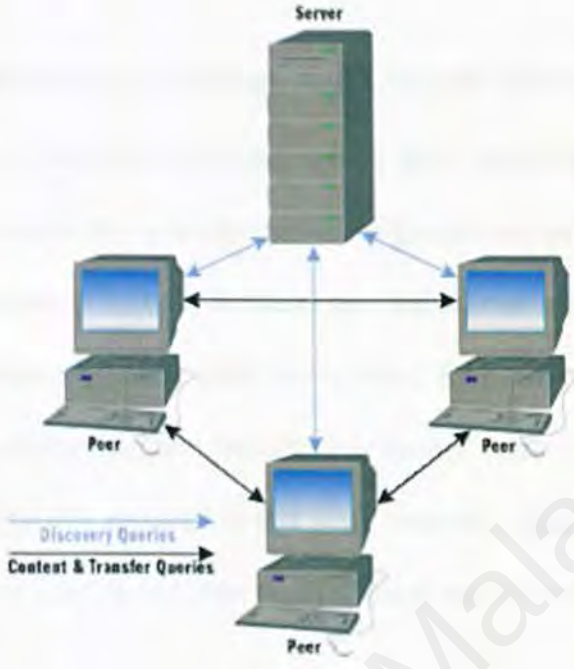


Figure 2.2: P2P with a simple discovery server

2.2.3 Peer-to-Peer with a Discovery and Lookup Server

This model, similar to the one in the above figure, extends the discovery server so that it also includes content lookup services. In this case, besides registering with the discovery server, the peer application also uploads a list of its contents at regular intervals. When an application is looking for some particular content, it queries the central server rather than sending a query to each client. The central server then responds with a list of the clients that contain the requested content and the peer application can contact those clients directly to retrieve the content. Quite often this approach will scale better than the previous options because it reduces the number of queries going over the network. However, this saving will

incur a cost on the server. Servers are now more involved in the process of content sharing and the peer's demands will use significant resources.

2.2.4 P2P with a Discovery, Lookup, and Content Server

In this kind of model, the peers also upload their content onto the server. Thus the clients can now directly query the server for the contents and can download it from him. This approach effectively becomes the client/server model because the peers are no longer contacting other peers for content. Each peer registers with a server, queries it for information, and transfers any desired content down from the server. The problem with this approach is that when content is downloaded from all of the clients, the server quickly becomes the bottleneck and is easily overwhelmed by the peers (clients).

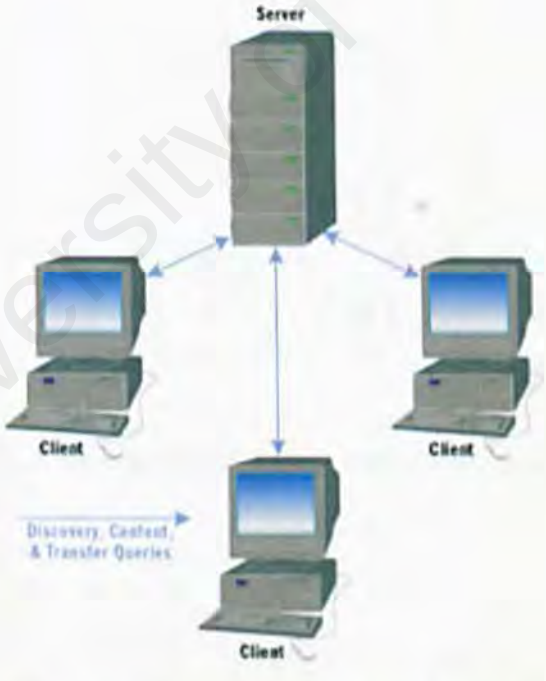


Figure 2.3: P2P with a discovery, lookup and content server

2.3 Analysis the Existing System

P2P technology has been used for almost as long as there have been computer networks. The internet itself can be seen as a large P2P network, with data passing from router to router. In general, a P2P system can fall into one of two categories. The first is a full distributed (pure) P2P system. The second is a central server P2P system.

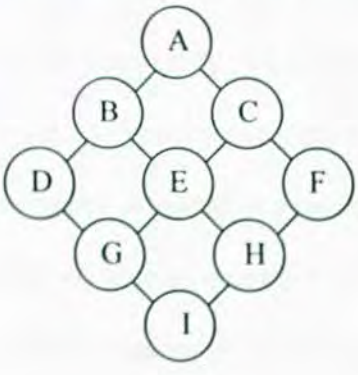


Figure 2.4: An example of pure P2P network

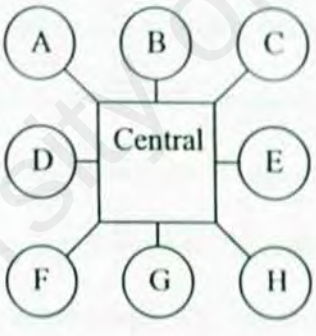


Figure 2.5: An example of central server P2P network

personal computer connected to a centralized server that interrogates the machine for the MP3 files that the machine's owner is willing to share.

Each computer in the Napster community knows about the Napster server to which it's directly connected. Napster users are largely visible and easily located by interrogation of the servers in the system. Although there is some level of hierarchy in the sense that a Napster server facilitates music searching, Napster still equally empowers each computer in the community. Your computer is unknown to anyone not directly connected to you unless you offer to share a file.

Queries into the community terminate at a Napster server that returns a list of the computers holding the music files of interest to the requester. Anonymity of a given member of the community is generally a moot point until a file is requested for download, at which time the computer is easily discovered, as the identity of the file's owner is usually attached to a file. Users in the system can bookmark a site that they find especially interesting and members can chat with one another.

The Napster servers act as central repository that lists all the files in the Napster community of every user machine. When a user request for a file gets a positive response from a node somewhere in the community, the two nodes connect directly and transfer the file. Interestingly, the actual music files move only from one peer to another. The centralized Napster server only holds information about the music files or metadata.

2.3.3 AIM (AOL Instant Messenger)

AIM [4] has the following main features:

- Instant messaging
- IM Images (transfer of inline images in Instant message conversations)

- Voice chat
- Game requests
- File transfers
- File sharing

Instant Messaging

Instant messaging is simply the passing of HTML-encoded clear text messages from one user to another, via a central BOS server. The message is not encrypted and is always routed over the Internet.

IM Images

IM images are sent via a direct connection with another peer. A request is sent to the BOS server and is relayed to the target user. The request packet for direct connection contains the TCP/IP address and port information of the requester. If accepted, the target of the request listens for an incoming request on port 4443 and a conversation begins between peers. These direct connections reveal the IP address of each participant.

Voice Chat

A direct connection must also be used for Voice Chat. Like in IM Images, data is transferred directly between peers. In this case it is sound data instead of images.

Game Requests

Game requests are simply requests for remote users to execute certain external programs, usually games. During game requests, no direct connection is made with peers via AIM. If the external application or game requires a direct connection, one may be set up. This is beyond the scope of AIM.

File Transfers

File transfers are very similar to image transfers in that a direct connection is established. However, once a file transfer is complete the direct connection is closed. As in IM Images, a BOS server relays a request packet to another user. When the recipient accepts the connection, a TCP port is opened to accept the incoming file. By default, this TCP port is 5190; however it is possible for the user initiating the file transfer to select any available port by nominating it in the File Transfer option dialog. The default security option is to allow file transfers from all users after displaying an accept file dialog box. These options can be configured in File Transfer Options in Preferences.

File Sharing

File sharing in AIM is a method that allows a user to browse a selected directory structure and to download files. File sharing is optional and must be enabled before any sharing can take place. The connection method for file sharing is the same as for a regular file transfer. The initiator sends a request packet to the target via the BOS server. After the target client accepts the request, the initiator begins listening on port 5190 (this can be changed in the File Sharing options dialog) and the target

sends the file list information. All file transfers are carried out over the same connection. If the shared directory is empty, the connection attempt will fail.

2.3.4 .NET Messenger

.NET Messenger [6] has the following main features:

- Instant messaging
- Voice/video chat
- Application sharing
- File transfers
- Remote Assistance
- Whiteboard

Instant Messaging

Instant messaging is simply the passing of HTML-encoded clear text messages from one user to another, via a central MSGR server. The message is not encrypted and is always routed over the Internet. Messages are sent to the server via TCP port 1863.

Voice/Video Chat

A direct connection must be used for Voice/Video Chat. This data is transferred via a UDP connection to ports 13324 and 13325.

Application Sharing

Application sharing gives a remote user access to programs installed on a computer. Optionally, a user can give control of a program to a remote user. If a user accepts

the invitation to share an application, the initiating user may select which of their own programs they wish to share with the other user. To achieve application sharing, a direct connection is established between clients over the TCP port 1503.

File Transfers

File transfers are very similar to image transfers in that a direct connection is established. However, once a file transfer is complete the direct connection is closed. File transfers in .NET Messenger require a direct connection to be made between two clients. At first, the user who wishes to send a file initiates a request, which passes through a .NET server and is received by another client. The client that initiated the file transfer listens for the receiver on port 6891/TCP.

Remote Assistance

Windows XP Professional and Home Editions contain the Remote Assistance utility, which allows a remote user to control another computer. The Remote Assistance feature in .NET Messenger launches this utility.

Whiteboard

Whiteboard sharing is a way to share a Microsoft Paint document over a direct connection. It is identical to Application Sharing. Starting a whiteboard session with another user is a shortcut of invoking Application Sharer, then selecting Microsoft Paint as the application to share.

2.3.5 Yahoo! Messenger

Yahoo! Messenger [7] has the following main features:

- Instant messaging
- Voice/video chat
- File transfers
- File sharing

Instant Messaging

Instant Messaging is simply the passing of HTML-encoded clear text messages from one user to another, via a central server. The message is not encrypted at all and is routed over the Internet in all cases. Messages are sent to the server via TCP port 5050.

File Transfers

File transfers require a direct connection between peers on port 80/TCP. A request packet is relayed via the central Yahoo! Server to another user. The sender listens on port 80/TCP for the recipient to accept and connect back, to receive the data. The service uses the common HTTP protocol for file transfers. The end user is able to configure the port that the Yahoo! Client listens on for file transfer connections.

File Sharing

File sharing in Yahoo! is a method that allows a user to browse a selected directory structure and to download files. File sharing is optional. However, it is enabled by default. The connection method for file sharing is the same as for a regular file

transfer. The initiator sends a request packet to the target via the Yahoo! server. After the target client accepts the request, the initiator listens on port 80 (this can be changed in the File Sharing options dialog) and the target sends the file list information. Transfers are carried out over the same connection. The default directory to be shared by Yahoo! Messenger is created by the installation program and is empty by default. This directory can be changed, and if the directory is empty, the connection attempt will fail.

The levels of user security available for File sharing are:

- Deny all
- Allow from all with Accept File dialog box
- Allow from people on "Buddy List" (no dialog box, automatic file transfer)

2.3.6 ICQ

ICQ [5] has the following main features:

- Instant messaging
- Voice/video chat
- File transfers
- File sharing

Instant Messaging

Instant messaging is simply the passing of HTML clear text messages from one user to another. The message is not encrypted and is always routed over the Internet. Messages are sent via TCP port 3570.

Voice/Video Chat

A direct connection must be used for Voice/Video Chat. This data is transferred via a UDP connection to port 6701.

File Transfers

File transfers require that a direct connection is established. However, once a file transfer is complete the direct connection is closed. To begin the file transfer, a request packet is sent via the standard Instant Message method to another user. After the user accepts the connection, the receiving client opens a TCP port on 3574 to accept the incoming file. The remote user must accept all file transfers by clicking Accept in a file transfer request dialog.

File Sharing

File sharing in ICQ is a method that allows a user to browse a selected directory structure and to download files from that directory structure. File sharing is disabled by default. It must be enabled before any sharing can take place. The connection method is similar to a regular file transfer. The initiator will send a request packet to the target. The target does not need to accept this request as long as the initiator is on their ICQ list. The initiator then connects to the remote system's TCP port 7320 and the target sends the file list information. All file transfers are carried out separate connections that seem to happen on randomly selected TCP ports. The default directory to be shared is created by the ICQ installation program and may be changed.

2.3.7 How Gnutella Works

Gnutella does not host any files, nor does it compile any central indexes. That is because there are no central servers in the Gnutella network. It is true P2P; the network consists of nothing but peers, all running some flavour of Gnutella-compatible software.

Officially, Gnutella is described as a decentralized file location technology, not a file sharing technology. This is an important distinction. The core Gnutella technology does nothing more than search for files. Only after a file is found can two Gnutella peers connect to each other and transfer the file from one peer to another.

Gnutella lets you search for and trade any type of file, not just MP3s. If you do a search for Britney Spears, for example, you will find MP3s of all her songs, along with photographs (in JPEG format), video clips (in MPEG, AVI, or QuickTime formats), and who knows what else. Pick the file you want, connect to the computer where that file is stored, and copy away.

Creating the Network

As stated in one of the many Gnutella FAQs circulating around the Web, Gnutella was designed to "create self-perpetuating networks that grow independent of any single company's involvement."

In other words, it is supposed to grow on its own.

The way it works is that you drop your computer into a pool of other computers. You connect to the computer that happens to be closest to you (virtually, not physically), and thus join the network. Your computer, then, becomes a new network node—and the Gnutella network expands as more nodes become connected.

The node that you first connect to is typically connected to several other nodes, simultaneously. That means, from your first entry into the network, you are linked to several other computers. Then, since those computers are linked to several other computers each, you are also linked (once or twice or three times removed) to all those other computers. In kind of a "six degrees of separation" way, you theoretically have access to every other node on the entire Gnutella network.

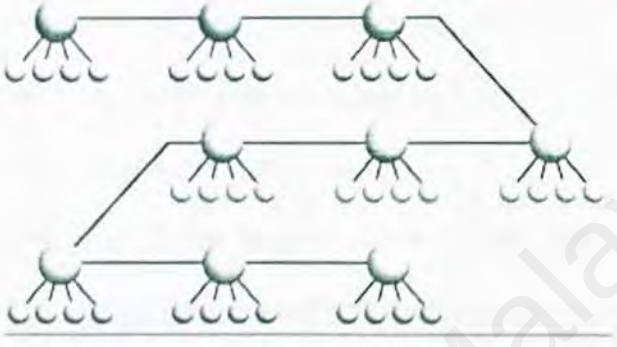


Figure 2.6: The topology of Gnutella Network

Interestingly, the Gnutella protocol allows for the creation of a relatively efficient backbone from all these different nodes. This is important, as some computers connect to the network at broadband speeds and others at slower dial-up speeds. Obviously, a more efficient network would divert more traffic to the higher-speed nodes and relegate the lower-speed nodes to the fringes of the network. That is just what Gnutella does; the higher-speed nodes migrate (virtually, of course) to the center of the network and carry more traffic, while the lower-speed nodes are shuffled to the edges of the network, where they would not be overburdened by (or slow down) excess traffic.

Searching for Files

Gnutella searches for files the same way you search for information in the real world. In the real world, if you need information, you might start by asking a few friends. If they do not know the answer, they will ask their friends—who will ask their friends, and so on, until somebody comes up with the information you need.

With the Gnutella network, when you initiate a file search, your query is first sent to the nodes to which you are most immediately connected. Those nodes pass your search along to the nodes they are connected to, and so on, until your search is propagated over a large portion of the network.

In most cases, you do not have to search all the way to the end of the network to find a file. Most queries are fulfilled within the first few hops, so you can click the "stop" button and keep your search from getting too far a field. Gnutella will also stop your search eventually, based on a time-to-live (TTL) number. Your query is initially assigned a TTL of 7, which dictates the maximum number of hops it can make. So after your query has been passed from node-to-node-to-node seven times, the TTL is up and the query ceases to propagate.

That does not mean you only search seven nodes. Remember that each node passes the query on to multiple nodes, and each pass—to multiple nodes—counts as a single hop. So a typical seven-hop "horizon" of a Gnutella query encompasses about 10,000 different nodes.

As files that match your query are found, information regarding each file is routed back to your computer. These results are displayed within your Gnutella client, in the form of a file list. (If you use wildcards in your search, your results list can get extremely long!) You can then choose to download any or all files on the list;

the transfers take place directly between your computer and the computer where the file is stored.

Anonymity and Survivability

Gnutella is almost—but not completely—anonymous. (Some Gnutella developers call this pseudoanonimity.) Just logging onto the network does not expose any information about you or your computer to any other node. When you execute a file search, there is no way that your search request can be traced back to your particular computer. But when you ultimately connect to another computer to transfer a file, your IP address is visible to the other computer. So, theoretically, an individual could collect the IP addresses of every Gnutella user—if, and only if, every user downloaded a file from that person's computer. Since that is unlikely, and since all other Gnutella activity is anonymous, there is no practical way for a central entity to monitor individual transactions.

And since individual transactions cannot be monitored, they also cannot be shut down. While it might be possible to block access from individual nodes, cutting off a few nodes doesn't significantly impact the network. In the quest to move data from point A to point B across the network, if one node goes dead, the data simply finds another, equally efficient, route. This means that the Gnutella network is highly survivable; if California gets hit by an earthquake, none of the other nodes will be affected.

Not only is Gnutella designed to survive network outages, it should also be able to survive any legal attack. Again, that's due to the total lack of any physical or virtual centralized server. There is nothing for the legal beagles to attack; shutting

down Gnutella would require shutting down every single node—which is both impractical and (since each node's IP address is hidden)

2.3.8 Algorithms

This section overviews three common P2P algorithms and then compares their implementations in a few P2P systems.

Centralized Directory Model

This model was made popular by Napster. The peers of the community connect to a central directory where they publish information about the content they offer for sharing. Upon request from a peer, the central index will match the request with the best peer in its directory that matches the request. The best peer in its directory that matches the request. The best peer could be the one that is cheapest, faster, or the most available, depending on the user needs. Then a file exchange will occur directly between the two peers. This model requires some managed infrastructure (the directory server), which hosts information about all participants in the community. This can cause the model to show some scalability limits, because it requires bigger servers when the number of users increase. However, Napster's experience showed that – except for legal issues – the model was very strong and efficient.

Flooded Request Model

The flooding model is different from the central index one. This is a pure P2P model in which no advertisement of shared resources occurs. Instead, each request from a

peer is flooded (broadcast) to directly connected peers, which themselves flood their peers etc., until the request is answered or a maximum number of flooding steps (typically 5 to 9) occur. This model, which is used by Gnutella, requires a lot of network bandwidth, and hence does not prove t to be very scalable, but it is efficient in limited communities such as a company network. To circumvent this problem, some companies have been developing "super-peer" client software that concentrates lots of the requests. This leads to much lower network bandwidth requirement, at the expense of high CPU consumption. Caching of recent search requests is also used to improve scalability.

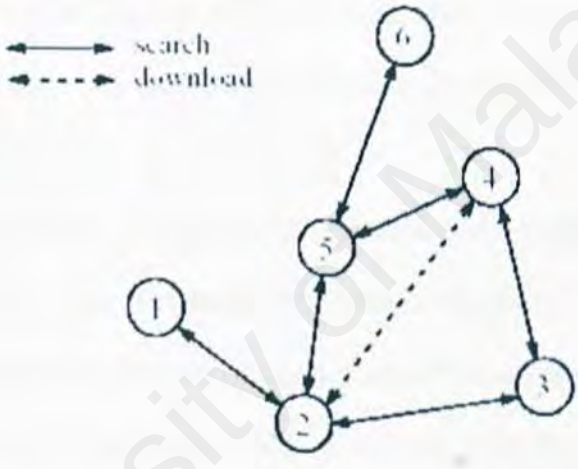


Figure 2.7: Flooded Request Algorithm

Document Routing Model

The document routing model, used by FreeNet, is the most recent approach. Each peer from the network is assigned a random ID and each peer also knows a given number of peers. When a document is published (shared) on such a system, an ID is assigned to the document based on a hash of the document's contents and its name. Each peer will then route the document's contents and its name. Each peer will then route the document towards the peer with the ID that is most similar to the document

ID. This process is repeated until the nearest peer ID is the current peer's ID. Each routing operation also ensures that a local copy of the document is kept. When a peer requests the document from P2P system, the request will go to the peer with the ID most similar to the document ID. This process is repeated until a copy of the document is found. Then the document is transferred back to the request originator, while each peer participating the routing will keep a local copy.

Although the document routing model is very efficient for large, global communities, it has the problem that the document IDs must be known before posting a request for given document. Hence it is more difficult to implement a search than in the flooded requests model. Also network partitioning can lead to an islanding problem, where the community splits into independent sub-communities, that do not have links to each other.

Four main algorithms have implemented the document routing model: Chord, CAN, Tapestry, and Pastry. The goals are to reduce the number of P2P hops that must be taken to locate a document of interest and to reduce the amount of routing state that must be kept at each peer. Each of the four algorithms either guarantees logarithmic bounds with respect to the size of the peer community, or argue that logarithmic bounds can be achieved with high probability. The differences in each approach are minimal, however each is more suitable for slightly different environment. In Chord, each peer keeps track of logN other peers (where N is the total number of peers in the community). When peer joins and leaves occur the highly optimized version of the algorithm will only need to notify logN other peers of the change. In CAN, each peer keeps track of only a small number of other peers (possibly less than logN). Only this set of peers is affected during insertion and deletion, making CAN more suitable for dynamic communities. However, the

tradeoff in this case lies in the fact that the smaller the routing table of a CAN peer, the longer the length of searches. Tapestry and Pastry are very similar. The primary benefit of these algorithms over the other two is that they actively try to reduce the latency of each P2P hop in addition to reducing the number of hops taken during a search.

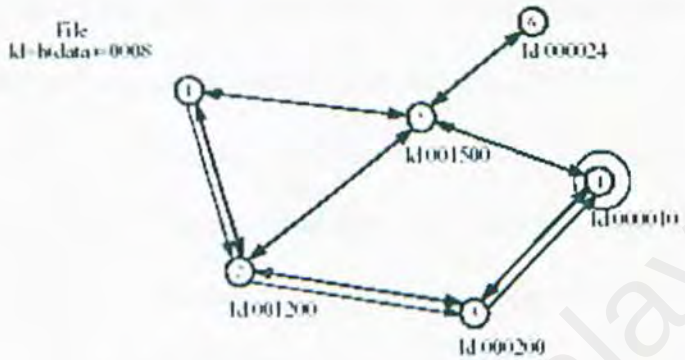


Figure 2.8: Document Routing Algorithm

Comparison of algorithms

The Chord algorithm models the identifier space as a uni-dimensional, circular identifier space. Peers are assigned IDs based on a hash on the IP address of the peer. When a peer joins the network, it contacts a gateway peer and routes toward its successor. The routing table at each peer n contains entries for logN other peers where the i-th peer succeeds n by at least 2^{i-1}. To route to another peer, the routing table at each hop is consulted and the message is forwarded toward the desired peer. When the successor of the new peer is found, the new peer takes responsibility for the set of documents that have identifiers less than or equal to its identifier and establishes its routing table. It then updates the routing state of all other peers in the network that are affected by the insertion. To increase the robustness of the algorithm, each document can be stored at some number of successive peers.

Therefore, if a single peer fails, the network can be repaired and the document can be found at another peer.

CAN model the identifier space as multidimensional. Each peer keeps track of its neighbours in each dimension. When a new peer joins the network, it randomly chooses a point in the identifier space and contacts the peer currently responsible for that point. The contacted peer splits the entire space for which it is responsible into two pieces and transfers responsibility of half to the new peer. The new peer also contacts all of the neighbours to update their routing entries. To increase the robustness of this algorithm, the entire identifier space can be replicated to create two or more "realities". In each reality, each peer is responsible for a different set of information. Therefore, if a document cannot be found in one reality, a peer can use the routing information for a second reality to find the desired information.

Tapestry and Pastry are very similar and are based on the idea of a Plaxton mesh. Identifiers are assigned based on a hash on the IP address of each peer. When a peer joins the network, it contacts a gateway peer and routes toward the peer in the network with the ID that most closely matches the its own ID. Routing state for the new peer is built by copying the routing state of the peers along the path toward the new peer's location. For a given peer n, its routing table will contain i levels where the i-th level contains references to b nodes (where b is the base of the identifier) that have identifiers that match n in the last i positions. Routing is based on a longest suffix protocol that selects the next hop to be the peer that has a suffix that matches the desired location in the greatest number of positions. Robustness in this protocol relies on the fact that at each hop, multiple nodes, and hence multiple paths, may be traversed.

2.4 Introduction to Microsoft .NET

Microsoft .NET [1] is software that connects information, people, systems, and devices. It spans clients, servers, and developer tools, and consists of:

- The .NET Framework programming model that enables developers to build web-based applications, smart client applications, and XML Web services applications which using standard protocols such as SOAP and HTTP.
- Developer tools, such as Microsoft Visual Studio .NET, which provide a rapid application integrated development environment for programming with the .NET Framework.
- A set of servers, including Microsoft Windows 2000, Microsoft SQL Server, and Microsoft BizTalk Server, that integrates runs, operates and manages XML Web services and applications.
- Client software, such as Windows XP, Windows CE, and Microsoft Office XP, that helps developers deliver a deep and compelling user experience across a family of devices and existing products.

2.4.1 The .NET Framework

The .NET Framework is the programming model of the .NET environment for the building, deploying, and XML Web services. It manages much of the plumbing, enabling developers to focus on writing the business logic code for their applications. The .NET Framework includes the common language runtime and class libraries.

2.4.2 Common Language Runtime

The common language runtime is responsible for run time services such as language integration, security enforcement, memory, process, and thread management. In addition, it has a role at development time when features such as life-cycle management, strong type naming, cross-language exception handling, dynamic binding, and so on, reduce the amount of code that a developer must write to turn business logic into a reusable component.

2.4.3 Class Libraries

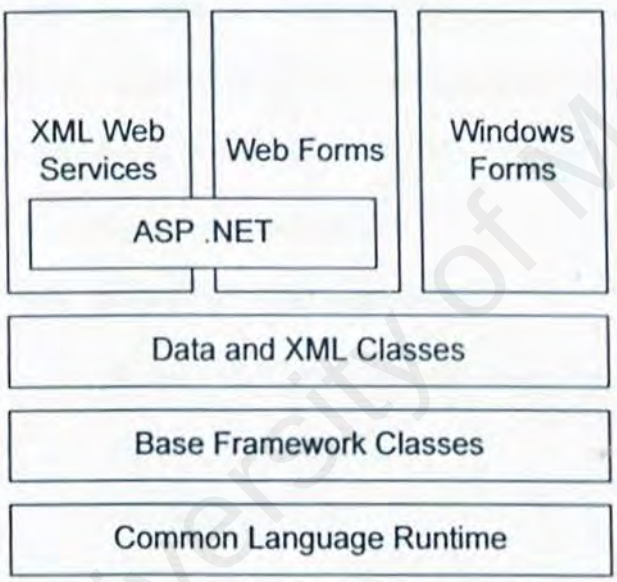


Figure 2.9: Class libraries in the Microsoft .NET framework

Base classes provide standard functionality such as input/output, string manipulation, security management, network communications; thread management, text management, user interface design features and other functions. The Microsoft ADO .NET data classes support persistent data management and include SQL classes support persistent data management and include SQL classes for manipulating persistent data stores through a standard SQL interface. XML classes

enable XML data manipulation and XML searching and translations. The Microsoft ASP .NET classes support the development of Web-based applications and XML Web services. The Windows Forms classes support the development of Windows-based smart client applications. Together, the class libraries provide a common, consistent development interface across all languages supported by the .NET Framework.

As you would expert from an object- oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use .NET Framework to develop the following types of applications and services:

- Console applications
- Scripted or hosted applications
- Windows GUI application (Windows Forms)
- ASP .NET applications
- XML Web services
- Windows services

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI Development. If you write an ASP .NET Web Form application, you can use the Web Forms classes.

2.4.4 Client Application Development

Client applications are the closest to a traditional style of application in windows-based programming. These are the types of applications that display

Windows or forms on the desktop, enabling a user to perform a task. Client application includes applications such as word processors and spreadsheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus, buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers.

Another kind of client application is the traditional ActiveX control (now replaced by the managed Windows Forms control) deployed over the Internet as a Web page. This application is much like other client applications: it is executed natively, has access to local resources, and includes graphical elements.

In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft Visual Basic. The .NET Framework incorporates aspect of these existing products into a single, consistent development environment that drastically simplifies the development of client applications.

The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command Windows, buttons menus, toolbars, and other screen elements with the flexibility necessary to accommodate shifting business needs.

For example, the .NET Framework provides simple properties to adjust visual attributes associated with forms. In some cases, the underlying operating system does not support changing these attributes directly, and in these cases the .NET Framework integrates the developer interface, making coding simpler and more consistent.

Unlikely ActiveX controls, Windows Forms controls have semi-trusted access to a user's computer. This means that binary or natively executing code can access some of the resources on the user's system (such as GUI elements and limited file access) without being able to access or compromise other resources. Because of code access security, many applications that once needed to be installed on a user's system can how be safely deployed through the Web. Your applications can implement the features of a local application while being deployed like a Web page.

2.4.5 Server Application Development

Server-side applications in the managed world are implementing through runtime hosts. Unmanaged applications host the common language runtime which allows your custom managed code to control the behavior of the server. This model provides you with all the features of the common language runtime and class library while gaining the performance and scalability of the host server.

2.4.6 Server –Side Managed Code

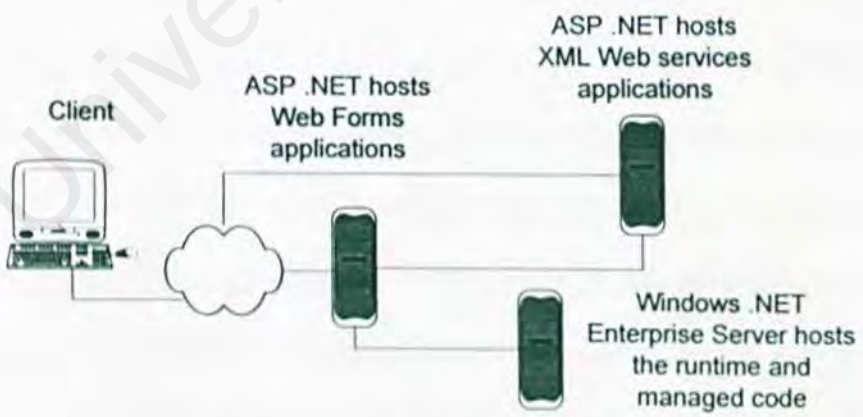


Figure 2.10: Server-side code management

ASP .NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP .NET is more than just runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects using managed code. Both Web Forms and XML Web Services use IIS and ASP .NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

XML Web Services, an important evolution in Web-based technology, are distributed, server-side application components similar to common Web sites. However, unlike Web-based applications, XML Web Services components have no UI and are not targeted for browsers such as Internet Explorer and Netscape Navigator. Instead, XML Web Services consist of reusable software components designed to be consumed by other applications, such as traditional client applications. Web-based applications are even other XML Web Services. As a result, XML Web Services technology is rapidly moving application development and the highly distributed environment of the Internet.

If you have used earlier versions of ASP technology, you will immediately notice the improvements that ASP .NET and Web Forms offers. For example, you can develop Web Form pages in any language that supports the .NET Framework. In addition, your code no longer needs to share the same file with your HTTP text (although it can continue to do so if you prefer). Web Forms pages execute in native machine language because like any other managed application, they take full advantage of the runtime. In contrast, unmanaged ASP pages are always scripted and interpreted. ASP .NET pages are faster, more functional, and easier to develop than unmanaged ASP pages because they interact with the runtime like any managed application.

The .NET Framework also provides a collection of classes and tools to aid in development and consumption of XML Web Services applications. XML Web Services are built on standards such as SOAP (a remote procedure-call protocol), XML (an extensible data format), and WSDL (the Web Services Description Language). The .NET Framework is built on these standards to promote interoperability with non-Microsoft solutions. For example, the Web Services Description Language tool included with the .NET Framework SDK can query an XML Web service published on the Web, parse it WSDL description, and produce C# or Visual Basic source code that your application can use to become a client of the XML Web service. The source code can create classes derived from classes in the class library that handle all the underlying communication using SOAP and XML parsing. Although you can use the class library to consume XML Web services directly, the Web Services Description Language tool and the other tools contained in the SDK facilitate your development effects with the .NET Framework.

If you develop and publish your own XML Web service, the .NET Framework provides a set of classes that conform to all the underlying communication standards, such as SOAP, WSDL, and XML. Using those classes enables you to focus on the logic of your service, without concerning yourself with the communications infrastructure required by distributed software development. Finally, like Web Forms pages in the managed environment, your XML Web service with the speed of nature machine language using the scalable communication of IIS.

2.4.7 High-level Statistics and Market Momentum

- The .NET Framework has been live since January 2002. The Beta release over 4 million developers world wide through 3.5 million CDs and 700,000 unique downloads.
- Compilers for over 20 programming languages are available for use with the .NET Framework.
- Dozens of components and controls are now available from third-party vendors.
- Microsoft is aggressively deploying .NET Framework-based applications today. MSN and the Microsoft .com Smart 404 are just a few of the many Microsoft applications already transitioned to the .NET Framework.

2.4.8 Rapid Development

- The multiple-language capability of the .NET Framework enables developers to use the programming language that is most appropriate for a given task and to combine languages within a single application. Support for the .NET Framework has been announced for over 20 commercial and academic programming languages.
- The component-based, plumbing-free design of the .NET Framework enables developers to write less code. The sample .NET Pet Shop, the .NET-based version of Sun's best-practice sample application, Java Pet Store, implements the same functionality as the Java2 Enterprise Edition (J2EE) version, but it uses one-third the code of the J2EE version.

2.5 Development Platform

Operating system (OS) is a platform that performs basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers.

Besides that, the OS ensures that different programs and users running at the same time do not interfere with each other. For security, OS will restrict the access of unauthorized users to the system. OS provides a software platform to allow application programs run on it.

The most popular operating systems currently are Microsoft Windows, Linux and Unix.

Currently, Microsoft Windows is the most favoured operating system in the market. A few versions of Microsoft Windows are now available, that are:

- Microsoft Windows 98
- Microsoft Windows NT
- Microsoft Windows 2000 Professional
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Data Center Server
- Microsoft Windows ME
- Microsoft Windows XP

The following section will emphasize on the Microsoft Windows XP Professional.

2.5.1 Windows XP Professional

Microsoft Windows XP Professional is the latest operating system for Windows family. It is the next version windows operating system for Windows 2000 and Windows Millennium. It build base on NT technology and Windows 2000 kernel which giving the user a more stable and reliable environment than previous version of Windows. User Interface in Windows XP has been redesigned to give user a new experience in Windows computing.

Features provided by Microsoft Windows XP Professional are similar to Microsoft Windows 2000 Professional.

Major enhancement and feature in Windows XP

- Intelligent User Interface
- Improved Networking and Communications
- Stranger Security Protection
- Internet Information Services 5.2
- Mobile Computing Support
- Enhanced File System
- Greater Application and Device Compatibility
- Basic Firewall Function

2.6 Development Technology

Visual Studio .NET is not part of the .NET Framework. However, it deserves mention in an introduction of the .NET Framework. Visual Studio .NET is an integrated development environment published by Microsoft for writing Windows programs. Visual Basic and any other language (such as Perl) that is integrated into the environment by a third-party.

Visual Studio .NET itself is a partially managed application and requires the .NET Framework to run. Visual Studio .NET is a very user-friendly and productive environment in which to write managed applications. It includes many helpful wizard for creating code, as well as useful features such as context coloring, integrated online help, auto completion and edit time error notification. But you do not need Visual Studio .NET to execute or develop managed software. It is important that you recognize Visual Studio .NET and the .NET Framework as different products.

The Framework is the infrastructure for managed code. The .NET Framework includes the CLR as well as other components that I will be discussing shortly. The .NET Framework also ships with an SDK (Software Development Kit) that includes command line compilers for C#, C++, Visual Basic and IL.

The Bottom line is that the Framework is all you need to develop applications such as C#, C++, Visual Basic, and so on. That being said Visual Studio .NET can increase your enjoyment and productivity significantly.

Visual Studio .NET is a development tools for building ASP Web applications. XML Web services, Windows applications and mobile applications. Different from the previously version, Microsoft Visual Studio .NET integrated

Visual Basic .NET, Visual C++ .NET, and Visual C# .NET into a single development environment (IDE) which allows them to share tools and facilitates in the creation of mixed-language solutions. In addition, these languages support the functionality of the .NET Framework, which provide access to key technologies that simplify the development of ASP Web Applications and XML Web services.

Major Features and Enhancement in Visual Studio .NET

- General Integrated Development Environment Tools.
- Application Templates.
- Support Extensible Markup Language (XML).
- Supports for new programming language – Visual C#.
- Drag and Drop function, enable developer to create a function without writing plenty of code.
- Database and Software modeling using Microsoft Visio.

2.6.1 Visual Basic .NET

Visual Basic .NET is the next generation programming language of Visual basic, is designed to be the easiest and most productive tool for creating .NET applications, including Windows applications, Web services and Web applications. Visual Basic .NET enables developers to create rich applications for Microsoft Windows in less time, incorporate data access from a wider range of database environment, create components with minimal code and build Web-based applications.

While providing the traditional ease-of-use of Visual basic development, Visual Basic .NET also allows optional use of new language features. Inheritance, method overloading, structured exception handling and free threading all make

Visual Basic .NET a powerful object-oriented programming language. Visual Basic .NET fully integrates with the .NET Framework and the Common Language Runtime, which together provide language interoperability, simplified deployment, enhanced security and improved versioning support.

2.6.2 Visual C# .NET

Visual C# .NET is the newest programming language in computer world. It pronounced as "C sharp". Visual C# .NET is designed to be a fast and easy way to create a .NET application. C# is an object-oriented language that very similar to C or C++. C# contains the best features of C++ but leaves out other features that does not required in .NET Framework such as typedefs and templates. As a result, developers familiar with C/C++ or Java can master the skill of C# easily. C# is fully integrated with the Microsoft .NET Framework and the Common Language Runtime(CLR). This and CLR such as language interoperability, garbage Icollection, enhanced security, and improved versioning support. C# can be used in developing a Windows application, Web application, Windows components or even XML Web services.

2.6.3 Visual C++ .NET

Visual C++ .NET enables developers to create powerful applications using the C++ language. With traditional unmanaged C++, developers can continue to create powerful C++ applications for Windows. With the new Managed Extensions for C++, developers can .NET-able their existing applications or create new .NET-enabled applications using the C++ language.

2.6.4 Visual J# .NET

Microsoft Visual J# .NET is a development tool for Java language developers who want to build applications and services on the Microsoft .NET Framework. Visual J# .NET is designed to take full advantage of the .NET Framework advantages and features. This includes ASP .NET, ADO .NET, Windows Forms, and XML Web services, as well as full cross-language integration. The syntax of J# .NET is similar to Java which enables Java developer to build the .NET application without re-training. The J# .NET is fully integrated with Microsoft Visual Studio .NET. This means the all the features and functions provided by Visual Studio .NET is fully utilized for J# .NET. Visual J# .NET joins more than 20 previously announced languages with its ability to target the .NET Framework and first-class XML Web services.

Visual J# .NET includes technology that enables customers to migrate Java language investments to the .NET Framework. Existing applications developed with Visual J++ can be easily modified to execute on the .NET Framework, interoperate with other .NET-connected languages and applications and incorporate new .NET functionality such as Microsoft ASP .NET, Microsoft ADO .NET, and Microsoft Windows Forms. Further, developers can use Visual J# .NET to create entirely new .NET-connected applications.

2.7 Database Management System (DBMS)

A database is a structured collection of data. To add, access, and process data stored in a computer database, a database server is needed. There are several database servers available currently such as the Oracle 9i, MySQL, and SQL Server

2.7.1 Oracle 9i

Oracle is the world leading Data Management System (DBMS) provider. Oracle's relational database was the world's first to support the Structured Query Language (SQL), now an industry standard, Oracle database is targets to high-end workstations and minicomputer as the server platforms such as UNIX and LINUX.

Oracle 9i is a database specifically designed as an Internet development platform, extending Oracle's long-standing technology leadership in the areas of data management, transaction processing and data warehousing to be the new medium of the Internet. Oracle 9i database also supplies a solid foundation for data warehousing.

Besides delivering the performance, scalability and availability required of Internet applications, Oracle 9i provides a fast and scalable data warehouse, with features that simplify operation and management. The Oracle 9i database incorporates numerous enhancements in query processing to deliver faster analysis to more users. Increased scalability and higher performance also result from server-based analysis functions, such as ranking and roll-up operations. Oracle 9i also has close integration with Windows 2000 and provides support for .NET environment using OLE DB driver.

Features in Oracle Database 9i:

- Oracle Enterprise Manager for data management and maintenance.
- Oracle Change Manager shows user how scheme changes will affect the database without interfering with production systems.
- Oracle Performance Manager for maintaining the health of the Server.
- Provide email and pager alerts for common database events or critical conditions, such as running low on table space.
- Fully support for XML and Java.
- SQL Plus for queries again database.
- Oracle Data Guard for data protection.
- Close integration with Windows 2000 and provide support for .NET environment using OLE DB driver.

2.7.2 MySQL

MySQL is a relational database management system. MySQL stores data in separate tables rather than putting all the data in a location. This adds speed and flexibility. The tables are linked by defined relations making it possible to combine data from several tables on request.

MySQL is a small, compact, easy to use database server, ideal for small and medium sized applications. It is client/server implementation that consists of a server and many different client programs. It is available on a variety of UNIX platforms, Linux, Windows NT, Windows 95/98 and Windows 2000.

Today MySQL is the most popular open source database server in the world with more than 2 million installations powering websites, data warehouses, business applications, logging systems and more. Customers such as Yahoo! Finance,

MP3.com, Motorola, NASA, Silicon Graphics, and Texas Instruments use the MySQL server in mission-critical applications.

2.7.3 Microsoft SQL Server 2000

SQL Server 2000 provides agility to your data management and analysis, allowing organizations to adapt quickly and gracefully to derive competitive advantage in a fast-changing environment. From a data management and analysis perspective, it is critical to turn raw data into business intelligence and take full advantage of the opportunities presented by the Web.

SQL Server 2000 is a fully Web-enabled database server, providing core support for Extensible Markup Language (XML) and the ability to query across the Internet and beyond the firewall.

Microsoft SQL Server 2000 provides the following features:

- Simplify the integration of back-end systems and data transfer across firewalls using XML.
- Connect to SQL Server 2000 databases and OLAP cubes flexibly, by using the Web with no additional programming.
- Ensure applications are secure in any networked environment, with role-based security and file and network encryption.
- Use and manage both structured and unstructured data, including searching through Microsoft Office documents.
- Enable the implement of merge, transactional, and snapshot replication with heterogeneous systems with SQL Server 2000.
- Automatic tuning and maintenance features enable administrators to focus on other critical tasks.

- User-defined functions, cascading referential integrity and the integrated Transact-SQL debugger allow users to reuse code to simplify the development process.
- Enterprise Manager for tables, created users, and set table and column permissions graphically.
- Query Analyzer for Database query and processing using SQL statement.
- Backup Wizard makes scheduling database backups.
- SQL Server Enterprise Edition can use up to 32 processors and 64GB of RAM.
- Able to use HTTP to send queries to the database.
- Database may be access by .NET application using ADO .NET.
- Fully integrated with Office 2000, Microsoft BizTalk Server and Microsoft Commerce Server.
- Support for XML format.
- Easy installation.

2.8 Network

Computer can play three roles on a network:

- Client
- Peer
- Server

A network computer acting as a client uses services provided by network servers but does not offer any services of its own. A networked computer acting as a peer performs as a client when a user is working at the computer while accessing network resources but also acts as a server by making a local resource available over the network to other computer.

A computer acting as a server on the network is typically a high-performance machine with more hardware resources than other computers. It provides services to other computers.

As just explained, there are two kinds of Windows networks: peer and server. On Windows peer networks, all computers can act as both peers and clients. When a computer is sharing a resource with another computer on a peer network, it is acting as a server, and when it accesses a resource provided by another computer on the network, it is performing the role of a network client.

2.8.1 Peer Network Features

Peer networks have many features that differentiate them from their server-based network counterparts. On a peer network, all users log on to the computer where they will work. There is no central account server to validate user identity or determine whether the user has permission to access the network. As long as the

computer has the proper hardware and software installed, the user can at least view network resources.

Every computer on a peer network, therefore, maintains its own information about the users who log on to that computer. On a peer network with five computers, all five computers maintain separate and distinct user account information.

Advantages

Security information is defined and stored locally on each computer. If one primary person is responsible for administration of the network, that person must physically or remotely visit every computer on the network to administer security, define new users, and perform other types of network administration. On server-based networks, all this work can be performed on a single centralized network computer.

One big advantage of peer networks is that no extra costs in terms of software are required to set up a peer network, and the cost of hardware is limited to cabling, network adapter cards, and an optional hub. Setting up a client-server network requires the purchase of additional networking software and dedicated server computers as well. Peer networks are relatively easy to set up compared to server networks.

Client-server networks require a dedicated network administrator who is technically knowledgeable and experienced. Peer networks, on the other hand, do not usually have a dedicated administrator.

In a peer network users are generally responsible for managing their own computer. This leaves the responsibility of setting up shared resources and applying security to the individual network user.

Therefore, the stability of the network can depend greatly on the cooperative participation of its user. Another option is to appoint a single person to perform most of the network administrative duties so that individual users can be left to utilize the network to get things done.

Overall, the total cost of setting up and maintaining a peer network of two to ten workstations is considerably less than that for setting up a client-server network of the same size. You can set up a peer network to emulate a client-server network by using powerful peer computers that are dedicated to providing services to other computers on the network. For example, a company might set aside one computer on the peer network that no one uses to act as a workstation; the computer is dedicated exclusively to providing a service to the network. Using a computer in this manner on a peer network results in the creation of a dedicated server.

Disadvantages

Peer networks have some disadvantages compared to client-server networks. Additional load is placed on peer computers that share their resources over the network. The size of the network is typically limited to ten computers. And because peer networks have no central administration, the possibility of disorganization and confusion is higher for peer networks.

Peer networks have decentralized security and usually depend on trusted users and good physical security to keep data safe. When responsibility is placed in the hands of users, it becomes more difficult to enforce standards and policies. On a client-server network, the network administrator can enforce operating-system policies as part of the network security model.

2.8.2 Client-Server Network Features

In contrast to peer networks, client-server networks are almost always more reliable, better performing, and more secure. But all these advantages come at a price, which includes dedicated high-end hardware, additional software purchases, and compensation for at least one dedicated network administrator.

A client-server network dedicates at least one computer to provide centralized security and user account management. Centralized security allows for single password access to all network resources, as opposed to maintaining multiple user accounts on peer networks. As such, this type of network can prevent user access to the network through logon authentication. Because of dedicated server equipment, individual computers are relieved of the burden of providing network services. This might mean that users' computers can run with lower levels of hardware, such as less memory and smaller hard drives.

As client-server networks grow in size, they are able to more effectively manage large numbers of users. Client-server computers can be configured and optimized for specific functions to provide the best possible service. Removing network administration from users and placing the responsibility in the hands of a few administrators makes enforcement of network policies much easier.

Of course, client-server networks have their drawbacks as well. They are always more expensive than their peer counterparts. They require dedicated hardware, software, and at least one network administrator.

Table 2.1: Windows peer and client-server network

Windows Peer Networks	Windows Client-Server Networks
Lack central security	Maintain centralized security
Are suitable for 2 to 10 computers	Are suitable for any-sized network
Depend on distributed user-level security, which uses individual passwords to protect resources	Provide both share-level and user-level security to protect network resources
Are not optimized for sharing resources; user workstations that share resources can take performance hits	Have dedicated hardware that supports client-server resources and can be configured to provide optimal network performance
Are less expensive – no extra cost in server hardware or software	Are more expensive – require dedicated hardware and additional software
Have no central administrator	Require at least one dedicated administrator
Depend heavily on physical security	Have excellent built-in security

2.9 Chapter Summary

In this chapter, we are introduced about P2P and Microsoft .NET technologies. Different reviews have been done on multiple development aspects for P2P application. The aspects in the reviews are the types of operating systems, development tools, database systems, and networks.

The operating system used to develop the P2P application is Microsoft Windows XP Professional. Microsoft Visual Studio .NET is the development tool used to develop the P2P application. The database system for the P2P application is Microsoft SQL Server 2000.

Chapter 3 Methodology

3.1 Introduction

System analysis is a process of gathering and interpreting facts, diagnosing problems as well as using the information to recommend improvement to the system. In this process, the information gathered has provided alternative strategies to develop this system.

Through system analysis, the programmer may add, delete and modify system components toward the goal of improving the overall system. The information gathered during this phase has provided alternative strategies to develop this system. Although software design can be identified and defined as a distinct activity, it must be compatible in both concept and implementation with essential development activities such as analysis and coding that precede or follow it. Through this phase also, the programmer can types of functional requirements and nonfunctional requirements for the system.

The system development methodology is a series of operations and procedures that are used to build up a system. The system development life cycle can be used to represent and define the methodology of the system development.

Every system development process includes flow of the system requirement to the finished product. The figure below shows the system development process model for a general system development.

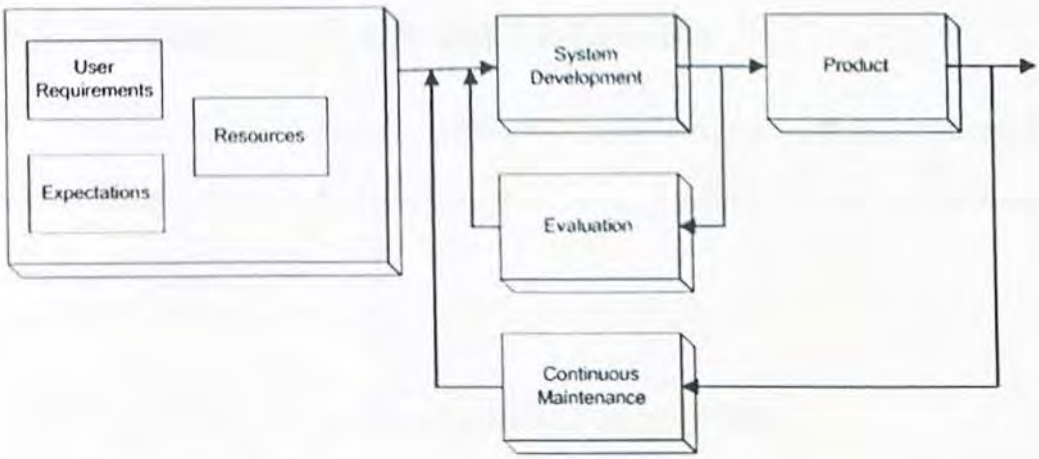


Figure 3.1: System development process model

3.1.1 Objectives of System Analysis

Here are some of the objectives for the analysis:

- Determine type of methodology to be used
- Study the problem faced by the user and find out the best solution to reduced it
- Acquire knowledge on how this system will be developed with the new emerging technology
- Choose the development tools for the new system among different types of tools that have been studied in Chapter 2
- Identify the major modules to be included in the system and what are the modules that are feasible to develop and the tools needed to have in order to develop it

3.2 System Development Methodology

In this section, the type of software development process model to be used will be discussed. The techniques used to define the requirements will also be used to be decided.

3.2.1 Type of Software Development Process Model

Many process models are described in the software engineering literature [2][3]. Some are prescriptions for the way software development should progress, and others are descriptions of the way software development is done in actuality. In theory, the two kinds of models should be similar or the same, but in practice, they are not. Every software development process model includes system requirements as input and a delivered product as the output.

There are several types of software development process models that are commonly used by most organizations. These models are:

- Waterfall Model
- Waterfall Model with Prototyping
- V Model
- Prototyping Model
- Operational Specification Model
- Transformational Model
- Phased Development: Increments and Iterations
- Spiral Model
- Rapid Development

3.2.2 Rapid Application Development

Rapid Application Development (RAD) is a newest approach to developing information system, which provides better and cheaper systems and more rapid deployment by having systems developers and end users work together jointly in the real time to develop systems. RAD grew out of the convergence of late 1990 and early 2000, and the ready availability of high powered computer based tools to support system development and easy maintenance.

RAD is an adaptation of traditional System Development Life Cycle (SDLC) but are shorted and combined to produce a more streamlined development technique. There are 6 phases in RAD methodology that are Business Modeling, Data Modeling, Process Modeling, Application Generation and Testing and Turnover.

Business and data modeling phases is focusing works on system functional and user requirement at the expenses of detailed business analysis and concern for system performance issue. The emphasis RAD is generally less on the sequence and structure of processes in the life cycle and more on doing different tasks in parallel with each other and on using prototype extensively. The iteration in the RAD life cycle is around to the design and development phases where the bulk of the works in a RAD approach take place.

According to James Martin, RAD can reduce a system in six months that would take 24 months to provide using the traditional systems development life cycle.

Below are the 6 phases in RAD consisting the specific tasks used to developing P2P application.

3.2.3 Project Development Phases

1. **Business Modeling**

   - An activity which is to identify the project problems, establishing project initiation plan, describing project scope and alternative.
   - Collect information from survey, interview, observation, throwaway prototype creation and study documents about the issues of the needs for the system.

2. **Data Modeling**

   - The information which collected in Business Modeling phase is defined into a set of data objects that's need to support the business process.
   - The attributes of each object are identified and the relationship between each object is defined. The system's process which contains Data Flow Diagram and Entity Relationship Diagram is used to define the attribute and relationship for the system development.

3. **Process Modeling**

   - This phase is to design physical database or file, design the internal program and process. The data object design in the data modeling phase are transformed to implement a business function. Processing description is created for adding, modifying, deleting or retrieving a data object.

4. **Application Generation**

   - In this phase, application is created using the fourth generation programming technique such as Visual Basic. The core part of systems development which are coding the process logic, transform the business logic into computer code to perform the real life works.

- The RAD process works to reuse existing component or create reusable components when necessary to reduce the overall development process duration.

5. **Testing and Turnover**

- The system developers and end users participate together to test the application system to ensure the requirement system has been fulfilled. Since RAD process emphasizes reuse, many of the components have already been tested. This reduces the overall testing time.
- Consist of written or other visual information about the application system, how it works and how to use it.

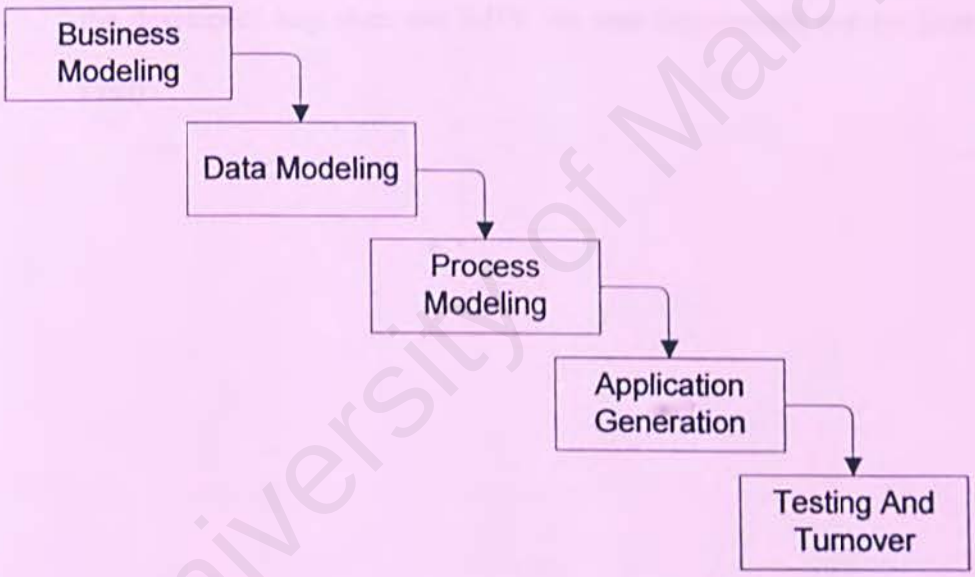


Figure 3.2: RAD model

The P2P application would be applying this methodology for system development process as the reason below:

- To decrease the time needed to develop

  RAD is emphasizing in reusable component, meaning that the component created will be reused in different module. As a result, less coding is need for

overall development process and this directly reduce the time needed for to develop the system.

- Reduce the expenditure cost in the project

  Since the time for overall development process is decrease by using the reusable concept, the expenditure cost in the project will also be reduced.

- Increase the usability of the system to fulfill the end users' needs.

  The requirement and information needed is collecting in Business Modeling phase and transform into appropriate design. Simple prototype is also created during this phase. As a result, better understanding of user requirement specification is produces. Users also participate in testing phase together with the developer. Any does not fulfill the user requirement can be defected easily.

3.3 Research and Analysis

There are varieties of technique that can be used to determine the requirements of the system or the users. These include the sampling and investigating hard data, interviewing, questionnaires, observing decision-maker behavior and office environment, and prototyping.

Effective and appropriate techniques must be used to define user requirements. A lot of information (requirements) is needed before development of a system can be done.

This information can be obtained from a variety of sources. Each source usually yields a different search method to get that information. Below is a few different search methods used to get that information.

The following sections will elaborate about the techniques used and will be used to do the research and analysis.

3.3.1 Library Research

A lot of materials in the library such as journal, conference and reference books offer a relatively concise information and format for research. During the process of the research and analysis, materials from the library have been helpful and provide a lot of solutions and references.

In the library, some books related to P2P application have been searched to have a better understanding for the P2P application system design and how does a P2P application be implementation effectively.

3.3.2 Internet Research

Notwithstanding that the library source has always been a good source of reference and research place, the Internet has grown to become a new resource of information. The advantages of using the Internet as the source of reference are that it is convenience, free and always up-to-date.

The Internet is the main resource of information in this throughout the whole development process of the P2P application system. Since the Internet is the huge an information warehouse, we can find and do a comparison for the resources gathered.

Surfing the net for sometimes, we can more understanding about the existing P2P application. The keywords used to search all the related information are:

1. Peer-To-Peer
2. Microsoft .Net Framework
3. Web services

3.3.3 Documents Room Research

The document room is a facility provided by the Faculty of Computer Science and Information Technology that consist of thesis, reference books and other reference materials to provide students, especially final year students as a place to search for materials.

The research and analysis in the documents room has been a great help to give a better view for the development of the P2P application system throughout the two semesters.

3.3.4 Interview

An interview is a directed conversation with a specific purpose that use a question end-to-end format. Seek the opinion of the person that are interviewing opinions may be more important and more revealing that facts. While gathering the requirements for this project, interviews were conducted to understand the problems of the existing system, to gather the requirements and features of the system from the administrators and users.

Interview is a technique of data collection where information is gathered by direct interaction with interviewees. Generally, the technique of interview can be divided into two categories which are structural and non-structural. Structural interviews will use planned and prepared questions as guidelines during the interviews. As an opposition, a non-structural interview will be performed with spontaneous questions.

Interview is a good technique to capture the precise requirements, information and problems of real life systems. An interview will help the interviewer to have a deeper understanding to the operations and limitations of the system in use from the interviewee.

The interview helps to provide a better and deeper understanding of the characteristic of the system, the pros and cons of the system and suitable functional requirements of the P2P application system to be developed.

3.4 Chapter Summary

This chapter introduces the methodology that will be used in the development, and system analysis for the P2P application system.

The methodology to be used in the development is the Rapid Application Development (RAD). RAD has been chosen as development methodology because it provides better and cheaper systems and more rapid deployment by having systems developers and end users work together jointly in the real time to develop systems. This model is used because it is able to decrease the time needed to develop, reduce the expenditure cost in the project, increase the usability of the system to fulfill the end users' needs, produce better software and it is systematic. This model is divided into 6 phases which are Business Modeling, Data Modeling, Process Modeling, Application Generation, Testing and Turnover.

The researches and analysis of the system are done using a few techniques which are library research, Internet research, document room research and interview. With these researches and analysis, the functional requirements and the non-functional requirements are determined.

Chapter 4 System Analysis

4.1 Introduction

A requirement is a feature of the system or a description of something the system is capable of doing in order to fulfill the system's purpose.

A software requirement definition is an abstract description of the services, which the system should provide, and the constraints under which the system must operate. There are two types of requirement analysis, functional requirement and non-functional requirement.

4.2 Functional Requirements

Functional requirements are statement of the services or functions that a system should provide how the system reacts to particular inputs, and how the system should behave in particular situations.

The main modules for the P2P application system are client module, server module and library module.

4.2.1 Client Module

Client module handles all the client functionalities including call request and call initiation.

4.2.2 Server Module

Server module handles all server functionalities like listening to connections and receiving messages.

4.2.3 Library Module

Library module handles all the Windows Form Controls, and also acts as a middle agent for the communication between client and server.

shall be taken into considerations. The usage of intuitive and meaningful menus and icons are also required.

4.3.2 Robustness

The modules for P2P application system will be wholly tested to ensure each module achieve its expectation. The modules will be integrated and system testing will be started after the integration. Any error that is discovered during system testing will be solved immediately. This will provide a robustness measure to the system expectations and reduce the possibility of failures during the implementation of the system.

4.3.3 Modularity

The coding for the P2P application system shall be implemented by using a modular approach so that the system can be modified or enhanced in the future. All the procedures, subroutines and methods in the program codes will be written in modular form. This will provide the future P2P application system developer to have a better understanding in the program codes. Furthermore, by using the modular approach will reduce the development time and prevent redundancy in the codes.

4.3.4 Reliability

P2P application system is a time critical system especially during the online messaging and files sharing are in progress. Thus, this system should be made reliable and should not cause unnecessary downtime to the overall system. The system should be set up based on the acceptable failure rate.

The system should be consistent when it is functioning. P2P application system should run smoothly even though many users are using the system simultaneously. The system should not produce dangerous or costly failures when it is used in a reasonable manner.

4.3.5 Response Time

All desirable information or downloads should be available to users at any point of time. The requirement for up-to-date information is also important.

During the progress of online messaging and files sharing, the system must be able to response quickly to users' action to prevent any time delay due to the ineffectiveness of the system. At the end of the online messaging and files sharing, users should be able to get the result feedback in a reasonable time (within a few seconds) to prevent the users to keep waiting.

4.3.6 Expandability

P2P application system should be build with the capability to accept enhancement. The system should be able to include new features, functionality and supports for larger databases.

4.3.7 Reusability

P2P application system should be build with program codes that are easy to maintain and modify. This will increase the reusability of the system to support other new functionalities or modules.

4.4 System Requirements

System requirements consist of two components which are:

- Software requirements
- Hardware requirements

The recommended system requirements are as follows:

Table 4.1: The hardware requirements

Hardware	Requirement
Processor	Pentium III 800MHz or higher or other equivalent processors
Memory (RAM)	Minimum of 256 MB (1 GB or more recommended)
Hard Disk Space	Minimum of 2 GB (8 GB or more recommended)
Others	Network Interface Card & other standard computer peripherals

Table 4.2: The software requirements

Software	Requirement
Operating System	Microsoft Windows 2000 Professional or higher
Web Server	Microsoft Internet Information Server 5.0 or higher
Database System	Microsoft SQL Server 2000 or Microsoft SQL Server Desktop Engine (MSDE)
Internet Browser	Microsoft Internet Explorer 5 or higher

4.5 Development Platform

In the consideration of operating system for the P2P application, Microsoft Windows XP Professional is chosen to be the development platform.

4.5.1 Microsoft Windows XP Professional

Windows XP is chosen to be the platform for the P2P application because of its ease of use and the stability of the system. Windows XP uses the 32-bit architecture that minimizes the chance of application failures and unplanned reboots. The stability of the system is an important feature that could affect the effectiveness and the fluency of the P2P application.

Another important consideration for the platform is that it should be user-friendly and user should be familiar with. Windows XP has gone through different versions from Windows 3.1 to Windows XP; users had been familiar with the Windows environment. Thus, users are expecting to be able to work well with the Windows environment.

Windows XP has the following enhanced features:

- **Business-level Reliability**

Windows XP delivers a new level of stability. So you can focus on your work. For example, in most cases, if one program crashes, your computer will keep running.

- **Advanced Performance**

Windows XP manages system resources efficiently, meeting the performance standards set by Windows 2000 and exceeding those set by Windows 98 Second Edition.

- **Remote Desktop**

Remote desktop allows you to create a virtual session and use your desktop computer from another computer running Windows 95 or later, giving you access to all of you data and applications even when you are not in your office.

- **New Task-based Visual Design**

Get to your most commonly used tasks quickly, thanks to a cleaner design and new visual cues.

- **Wireless 802.1x Networking Support**

Wireless 802.1x Networking Support provides support for secured access, as well as performance improvement for wireless networks.

- **Windows Messenger**

Windows Messenger is the easy way to communicate and collaborate in real time on your computer. You can see the online status of your contacts and choose to communicate with the through text, voice, or video with better performance and higher quality.

- **Encrypting File System**

Encrypting File System provides a high level of protection from hackers and data theft by transparently encrypting files with a randomly generated key.

- **Fast Resume From Hibernation Or Standby**

Save battery power when you are working on the road. With Windows XP your laptop can enter Standby or Hibernate faster, and it can start working again faster after resuming from Standby or Hibernation.

- **Help And Support Center With Remote Assistance**

In addition to a comprehensive set of documentation, Help and Support center in Windows SP includes Remote Assistance, which allows you to have a

friend or IT professional who is also running Windows XP remotely control your computer to demonstrate a process or help solve a problem.

- **System Restore**

If something goes wrong with your computer, you can revert the system back to a previous state.

4.6 Development Technology

4.6.1 Microsoft .NET Framework

Microsoft .NET Framework is a set of programming services designed to simplify the application development.

The advantages and benefits of Microsoft .Net Framework:

i. **Multiple Language Supports**

- Microsoft .NET such as Visual Basic .NET, Visual C++ .NET, Visual Jscript .NET, COBOL for Microsoft .NET, Fortran for Microsoft .NET, Pascal, SmallTalk for Microsoft .Net, Perl for Microsoft .NET as well as newest programming language that they familiar with, without any retraining program to suit themselves into new technology environment.
- Support language integration which means that developer can inherit from classes exceptions, and take advantage of polymorphism across different language.

ii. **Multiple Platform Support**

- .NET Framework can run on any versions of Windows, that's Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP as well as Windows CE. In future, it is expected that .NET will run other OS such as UNIX.

iii. **Improve Performance**

- .NET improves the performance of Web application. Previously, ASP code is interpreted when the code is running. However, in ASP .NET, all codes is compiled into MSIL and after that compiled into machine code by JIT.

- CLR can manage code efficiently than any previous version of runtime environment such as Visual Basic Runtime or Visual C++ Runtime.
- VeriTest, a software testing firm in United Stated found had participated in auditing. A Pet Shop System which develop using Microsoft .NET and Java 2 Enterprise Edition, found that Microsoft .NET Pet Shop performs over 10 times faster than Java Pet Shore.

iv. **Write Less Code**

- .NET Framework provides plenty of build-in control. Developers can use this control directly without writing their own function or control. This enables developers to focus on writing business logic rather than spent time in this pettiness task.
- VeriTest also found that Lines of Code to implement the Microsoft .NET Pet Shop was less than Sun's Java Pet Shop.

v. **Run More Reliable**

- Microsoft .NET Framework includes set of technologies to make application run more reliable. For example, memory threads and error handle managed by .NET Framework to ensure that memory leaks does not occur.

vi. **Simplified Deployment**

- To setup an application under .NET environment, just simply copy the source disk such as CD to the destination folder or directory using "xcopy method".
- Previously, the entire component must be register into registry before it can be run and operate correct. Under the .NET platform, all the developed component is does not require to register. The component will place in the

"bin" folder of each application. As a result, this simplified the way to install and remote an application from the system.

4.6.2 Microsoft Visual Studio .NET

Microsoft Visual Studio .NET is chosen as the Development Tool to develop this P2P application. Below is the benefit and advantages of Microsoft Visual Studio .NET:

- A complete set of development tools for building ASP Web application, XML Web services, desktop application and mobile application.
- Provide powerful WYSIWYG designer for Web pages, Intellisense HTML editing features and Style Sheet Editor.
- Provide an integrated development environment (IDE) for all supported languages such as Visual Basic .NET, Visual C++ .NET and Visual C++ .NET
- Developer able to use build-in ADO.NET object to create a data-driven application with Microsoft SQL Server, Oracle or any other XML-based data source.
- Smart editor which provide the automatic syntax error checking function.
- Simplified the way to create the Server-side component by using Drag and Drop concept.
- Provide a set of powerful tools for project and application setup and deployment.

- Provide debugging tools with cross-language support which help developer to find and fix error in the code. Developer also can use structured exception classes to build error handling into the application.
- Create and use XML Web service graphically.

This development tool is chosen because it is also following enhanced features:

- High-Performance Database Applications With XML

Use the built-in Microsoft ADO .NET tools to build database applications with Microsoft SQL server, Oracle, or any other XML-based data source. Take advantage of the Microsoft SQL Server, which natively supports XML for maximum interoperability.

- Rapid Application Development (RAD) For The Server

Visually compose middle-tier components using the Visual Component Designer, which enables developers to drag non-visual objects, such as message queues, timers, and event logs to a design surface from the Server Explorer, a new tool Windows that automatically discovers all necessary server-based resources.

- Complete Web Development Environment

Create Web solutions with any language, including Visual Basic and C#, using the shared Web Page Editor.

- Create Powerful Windows Applications

Visually construct Windows Forms applications by using the Forms Designer and visual inheritance to centralize in parent forms the common logic and user interface for the entire solution. Use control anchoring and docking to provide resizable forms automatically.

- Jump-Start the Development Process

Customize, automate and extend the IDE by using the Visual Studio .NET Environment.

4.6.2.1 Microsoft Visual Basic .Net

In this new version of Visual Basic, its fully support object-oriented feature. Now it is "true" Object-oriented Programming Language. Major Features and Improvement in Visual Basic .NET.

i. Inheritance

- Visual Basic .NET now supports Inheritance feature by allowing developer to define classes that serve as basic for derived classes.

ii. Exception Handling

- Structure Exception Handling now supported in Visual Basic .NET, using Try..Catch..Finally syntax.

iii. Overloading

- Overloading now supports in Visual Basic .NET. Overloading enable developer defines the properties, methods and procedures that have the same but using different data types.

iv. Constructor and Destructors

- Visual Basic supports constructor and destructors using Sub New and Sub Finalize procedures.

v. Multithreading

- Visual Basic .NET allows developer to write applications that can perform various tasks in a single process.

4.6.2.2 Microsoft Visual C# .NET

Visual C# .NET advantages:

i. Use C++ style syntax, developer who familiar with C++ and Java can easily implement the system using C# language.

ii. A Modem Object-Oriented Language, Support for properties, indexers, delegates, single and multi-dimensional arrays, advanced inheritance, attributes, versioning, and XML comments.

iii. Fully integrated with the .NET Framework and the Common Language Runtime, which together provide language interoperability, garbage collection, enhance security, and improved versioning support.

iv. Enables developers to build the next generation of Windows-based and Web-based applications.

v. Fully support for Extensible Markup Language (XML).

4.7 Database Management System (DBMS)

4.7.1 Microsoft SQL Server 2000

Microsoft SQL Server 2000 is one of the best Data Management System (DBMS) for Windows platform which development by Microsoft. It is fully supported in Microsoft .NET platform. It supports a high volume of concurrent users. It also provides easier way to backup and restore database.

Microsoft SQL Server 2000 has Enterprise Manager for tables, created users, and set table and column permissions graphically. It also has Query Analyzer for Database query and processing using SQL statement, Microsoft SQL Server 2000 able to use HTTP to send queries to the database. Database may be accessed by .NET application using ADO .NET. It is fully integrated with office 2000, Microsoft BizTalk Server and Microsoft Commerce Server. It supports for XML format and it is easy installation.

The reason why Microsoft SQL Server 2000 is chosen instead of Oracle 9i is because:

- SQL Server 2000 provides tight integration with Windows-base application compare to Oracle 9i which mainly support UNIX and Java platform.
- SQL Server 2000 is the .NET Enterprise Server product which means that if it fully supported in Microsoft .NET platform.
- Provide easier way to backup and restore database.
- Supports for XML which means that data can be retrieved as XML format instead of record set.
- Graphical User Interface for Database Management using Enterprise Server. This means that administrator can manage the database works such as create

database, edit data, update data or delete data using GUI method instead of using SQL statement.

- Support a high volume of concurrent users.
- Lowers the cost and complexity of distributed computing compare to Oracle 9i database which more expensive than SQL Server 2000. According to Microsoft, SQL Server 2000 is 3 times cheaper than Oracle 9i database.
- Strange security features. SQL Server 2000 automatically supports encryption of the data. The encryption strength is dependent upon the encryption capabilities authorized by the certificate installed for SQL Server and the cryptographic capabilities of the client and the server.

4.8 Network

4.8.1 Internet

Since the P2P application is an online system, Internet is the most suitable network type to be used in this project. Users from different places in the world can access to the P2P system if they have Internet access. The users of P2P application may have interactions with each other. Despite of the limitation of the geographical barrier, users all over the world can use this P2P application to communicate with each other.

4.9 Chapter Summary

P2P application system includes three modules. The modules are client module, server module and library module. The modules are further divided into smaller modules for ease of development.

Several non-functional requirements are defined for the P2P application system. These requirements are user-friendly, robustness, modularity, reliability, response time, expandability, and reusability.

Chapter 5 System Design

5.1 Introduction

System design is a process through which requirements are translated into a model or representation of the software that can be assessed for the quality before coding begins. On this stage, the requirements which are identified earlier are translated into system features and characteristics. The following designs have been considered:

- System Architecture
- User Interface Design

This phase will moves from what to do (the specification stage) to an interest in how to do it (the design stage).

5.2 System Architecture Design

A large system can be decomposed into sub-systems that provide some related set of services. Thus, system architecture design is the initial design process of identifying these sub-systems and establishing a frame for sub-system control and communication.

The system is structured into a number of principle sub-systems. Decomposing the system into a set of interacting sub-system is an important phase. Structure chart is used to depict the high level extraction of a specified system. The usage of structure chart is to describe the interaction between independent sub-systems.

The P2P application system consist three modules: client module, server module and library module.

Client Module

Client module handles all the client functionalities including call request and call initiation.

Server Module

Server module handles all the server functionalities like listening to connections and receiving messages.

Library Module

Library module handles all the Windows Form Controls, and also acts as a middle agent for the communication between client and server.

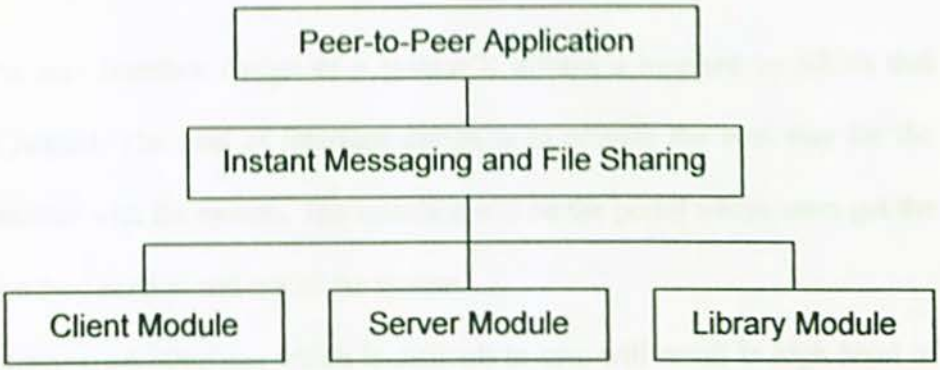


Figure 5.1: Main modules in the P2P application

5.3 User Interface Design

The user interface design of a system is always a measure by which that system is judged. The goal of interface design is to provide the best way for the users to interact with the system. The interface will be the portal where users get the information they need in and out of the system.

However, an interface, which is difficult to use, will result in high level of faults and errors, and may cause some system to be discarded, irrespective of its functionality. Thus, it is important to take into consideration the user's needs and preferences in the design of user interface.

In the P2P application system, the user interface design will base on the Graphical User Interface (GUI) approach. The goals of using the GUI approach are to provide user-friendly, easy and faster way for the user to interact with the P2P system.

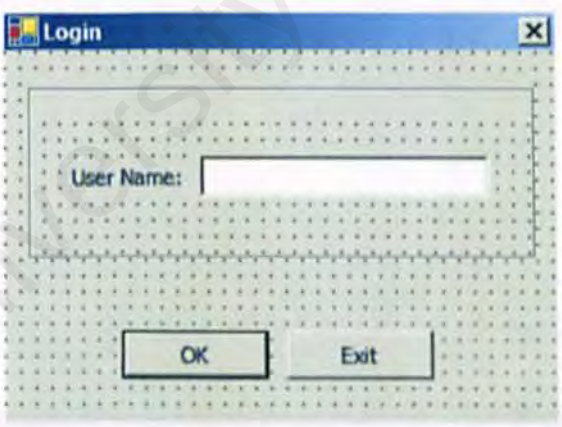


Figure 5.2: User login

Figure 5.3: User Interface for instant messaging and file sharing

Figure 5.4: Trace display

5.4 Chapter Summary

System design is a very important phase in developing a system regardless of whether the system is a new system that is to be built or a modification of an existing system where this phase will transform the model of a system on paper, to a real-life system. Thus every single part must be taken into account and careful consideration to avoid losing control in the process of system development.

The interface is designed in such way that users would not feel uneasy or boring when using the P2P application to communicate with other users, sending messages and sharing files. Every command button or label is clearly labeled.

Chapter 6 System Implementation

6.1 Introduction

Every Internet user is familiar with the basic model for instant-messaging application. Users log on to some sort of central authority, retrieve a list that indicates who else is currently online, and exchange simple text messages. Some messaging platforms include additional enhancements, such as file-transfer features and group conversations that can include more than two parties.

All current-day instant-messaging applications rely on some sort of central component that stores a list of who is currently online as well as the information needed to contact them. Depending on the way the system is set up, peers may retrieve this information and contact a chosen user directly, or they may route all activity through the central coordinator.

Conceptually, there are two types of applications in Student Talk .NET: the single server and the clients (or peers). Both applications must be divided into two parts: a remotable MarshalByRefObject that is exposed to the rest of the world and used for communication over the network, and a private portion, which manages the user interface and local user interaction. The server runs continuously at a fixed, well-known location, while the clients are free to appear and disappear on the network. Figure 6.1 diagrams these components.

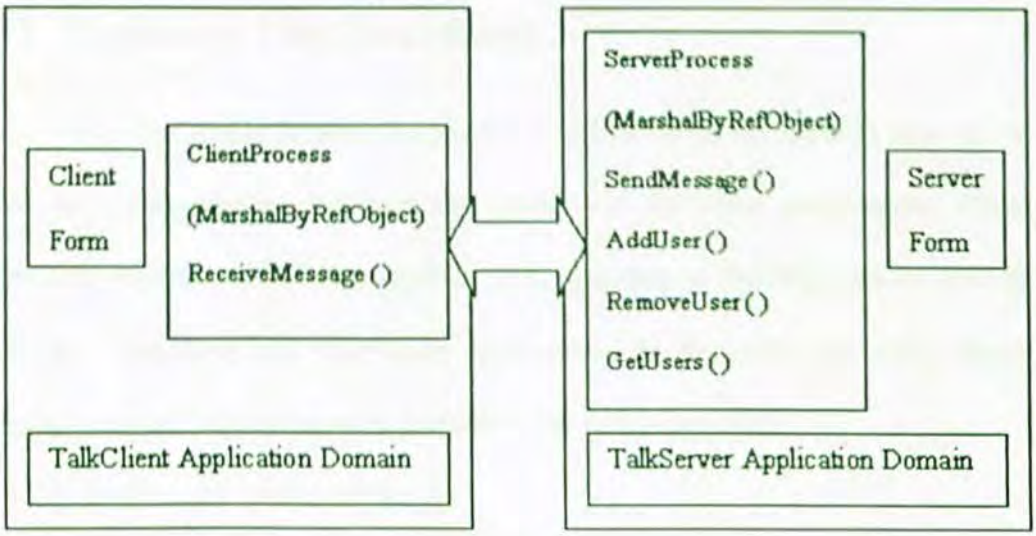


Figure 6.1: Components of the Student Talk .NET system.

In order for the server to contact the client, the client must maintain an open bi-directional channel. When a message arrives, the server notifies the client. This notification can take place in several ways – it might use a callback or event, or the server could just call a method on the client object or interface, which is the approach taken in the Student Talk .NET. Communication between these components uses TCP channels and binary formatting, although these details are easy enough to change through the configuration files.

One of the most important aspects of the Student Talk .NET design is the fact that it uses interfaces to manage the communication process. Interfaces help to standardize how any two objects interact in a distributed system. Student Talk .NET includes two interfaces: ITalkServer, which defines the methods that a client can call on the server, and ITalkClient, which defines the methods that the server (or another client) can call on a client. Before actually writing the code for the Student Talk .NET components, we will define the functionality by creating these interfaces.

6.2 Defining The Interfaces

The first step in creating the system is to lock down the methods that will be used for communication between the server and the client components. These interfaces must be created in a separate DLL assembly so that they can be used by both the TalkClient and TalkServer applications. In the code, this class library project is called TalkComponent. It contains the following code:

```
Public Interface ITalkServer

    ' These methods allow users to be registered and unregistered
with the server.
    Function AddUser(ByVal [alias] As String, ByVal callback As
ITalkClient) As Boolean
    Sub RemoveUser(ByVal [alias] As String, ByVal User As String)

    ' This returns a collection of currently logged in user names.
    Function GetUsers() As ICollection

    ' The client calls this to send a message to the server.
    Sub SendMessage(ByVal senderAlias As String, ByVal
recipientAlias As String, ByVal message As String)

    Function GetUser(ByVal [alias] As String) As ITalkClient

End Interface

Public Interface ITalkClient

    ' The server calls this to forward a message to the appropriate
client.
    Sub ReceiveMessage(ByVal message As String, ByVal senderAlias As
String)

    ' These methods are used to handle the file transfer process.
    Sub ReceiveFileOffer(ByVal filename As String, ByVal
fileIdentifier As Guid, ByVal senderAlias As String)
    Function TransferFile(ByVal fileIdentifier As Guid, ByVal
senderAlias As String) As Byte()
    Sub ClientDisconnected()

End Interface

' This delegate is primarily for convenience on some server-side
code.
Public Delegate Sub ReceiveMessageCallback(ByVal message As String,

ByVal senderAlias As String)
```

ITalkServer defines the basic AddUser() and RemoveUser() methods for registering and unregistering users. It also provides a GetUser() method that allows peers to retrieve a complete list of online users, and a SendMessage() method that actually routes a message from one peer to another. When SendMessage() is invoked, the server calls the ReceiveMessage() method of the ITalkClient interface to deliver the information to the appropriate peer.

6.3 Creating The Trace Component

Both the client and the server are depicted as Windows applications. For the client, this design decision makes sense. For the server, however, it is less appropriate because it makes the design less flexible. For example, it might make more sense to implement the server component as a Windows service instead of a stand-alone application.

A more loosely coupled option is possible. The server does not need to include any user-interface code. Instead, it can output messages to another source, such as the Windows event log. The Student Talk .NET server will actually output diagnostic messages using tracing code. These messages can then be dealt with in a variety of ways. They can be captured and recorded in a file, sent to an event log, shown in a console window, and so on. In the Student Talk .NET system, these messages will be caught by a custom trace listener, which will then display the trace messages in a Windows form. This approach is useful, flexible, and simple to code.

In .NET, any class can intercept, trace, and debug messages, provided it inherits from TraceListener in the System.Diagnostics namespace. This abstract class is the basis for DefaultTraceListener (which echoes messages to a TextWriter or Stream, including a FileStream) and EventLogTraceListener (which records messages in the Windows event log).

All custom trace listeners work by overriding the Write() and WriteLine() methods. The entire process works like this:

1. The program calls a method such as Debug.Write() or Trace.Write().

2. The common language runtime (CLR) iterates through the current collection of debug listeners (Debug.Listeners) or trace listeners (Trace.listeners).

3. Each time it finds a listener object, it calls its Write() or WriteLine() method with the message.

The solution used in this example creates a generic listener that forwards trace messages to a form, which then handles them appropriately. This arrangement is diagrammed in Figure 6.2.

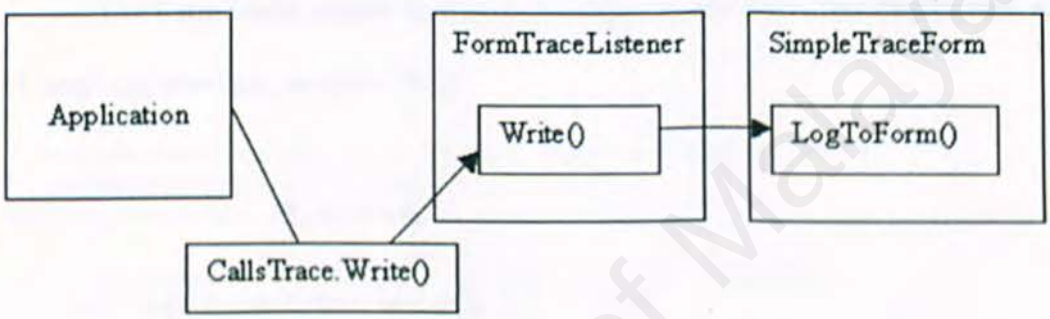


Figure 6.2: Forwarding trace messages to a form

The following is the outline for a FormTraceListener. This class is implemented in a separate class library project named TraceComponent.

```
' The form listener is a TraceListener object that
' maps trace messages to an ITraceForm instance, which
' will then display them in a window.
Public Class FormTraceListener
    Inherits TraceListener

    Public TraceForm As ITraceForm

    ' Use the default trace form.
    Public Sub New()
        MyBase.New()
        Me.TraceForm = New SimpleTraceForm()
    End Sub

    ' Use a custom trace form.
    Public Sub New(ByVal traceForm As ITraceForm)
        MyBase.New()

        If Not TypeOf traceForm Is Form Then
            Throw New InvalidCastException("ITraceForm must be used
on a Form instance.")
```

```
        End If

        Me.TraceForm = traceForm
    End Sub

    Public Overloads Overrides Sub Write(ByVal value As String)
        TraceForm.LogToForm(value)
    End Sub

    Public Overloads Overrides Sub WriteLine(ByVal message As
String)
        ' WriteLine() and Write() are equivalent in this simple
example.
        Me.Write(message)
    End Sub

End Class
```

The FormTraceListener can send messages to any form that implements an ITraceForm interface, as shown here:

```
' Any custom form can be a "trace form" as long as it
' implements this interface.
Public Interface ITraceForm

    ' Determines how trace messages will be displayed.
    Sub LogToForm(ByVal message As String)

End Interface
```

Finally, the TraceComponent assembly also includes a sample form that can be used for debugging. It simply displays received messages in a list box and automatically scrolls to the end of the list each time a message is received.

```
Public Class SimpleTraceForm
    Inherits System.Windows.Forms.Form
    Implements ITraceForm

    '(Designer code omitted.)

    Public Sub LogToForm(ByVal message As String) Implements
ITraceForm.LogToForm
        ' Add the log message.
        lstMessages.Items.Add(message)

        ' Scroll to the bottom of the list.
        lstMessages.SelectedIndex = lstMessages.Items.Count - 1
    End Sub

End Class
```

This approach is useful for Student Talk .NET server, but because it is implemented as a separate component, it can easily be reused in other projects.

6.4 The Coordination Server

After defined the basic building blocks for the Student Talk .NET system, it is time to move ahead and build the server. The TalkServer application has the task of tracking clients and routing messages from one user to another. The core of the application is implemented in the remotable ServerProcess class, which is provided to clients as a singleton object. A separate module, called Startup, is used to start the TalkServer application. It initializes the remoting configuration settings, creates and initializes an instance of the FormTraceListener, and displays the trace form modally. When the trace form is closed, the application ends, and the ServerProcess object is destroyed.

The startup code is shown here:

```
Imports System.Runtime.Remoting

Public Module Startup

    Public Sub Main()
        ' Create the server-side form (which displays diagnostic
information).
        ' This form is implemented as a diagnostic logger.
        ' That means that server trace messages are displayed
automatically,
        ' without requiring any specialized threading code.
        Dim frmLog As New TraceComponent.FormTraceListener()
        Trace.Listeners.Add(frmLog)

        ' Configure the connection, and register the well-known
object
        ' (ServerProcess) that will accept client requests.
        RemotingConfiguration.Configure("TalkServer.exe.config")

        ' From this point on, messages can be received by the
ServerProcess
        ' object. The object will be created for the first request,
        ' although you could create it explicitly if desired.

        ' Show the trace listener form. By using ShowDialog, we set
up a
        ' message loop on this thread. The application will
automatically end
        ' when the form is closed.
        Dim frm As Form = frmLog.TraceForm
        frm.Text = "Talk .NET Server (Trace Display)"
        frm.ShowDialog()
```

```
    End Sub
End Module
```

When start the server, the ServerProcess Singleton object is not created. Instead, it is created the first time a client invokes one of its methods. This will typically mean that the first application request will experience a slight delay, while the Singleton object is created.

The server configuration file is shown here. It includes three lines that are required if want to run the Student Talk .NET applications under .NET 1.1 (the version of .NET included with Visual Studio .NET 2003). These lines enable full serialization, which allows the TalkServer to use the ITalkClient reference. If you are using .NET 1.0, these lines must remain commented out, because they will not be recognized .NET 1.0 uses a slightly looser security model and allows full serialization support by default.

```
<configuration>
   <system.runtime.remoting>
      <application name="TalkNET">
         <service>
            <wellknown
               mode="Singleton"
               type="TalkServer.ServerProcess, TalkServer"
               objectUri="TalkServer"
            />
         </service>
         <channels>
            <channel port="8000" ref="tcp">
               <!-- If you are using .NET 1.1, uncomment the lines
below. -->
               <serverProviders>
                  <formatter ref="binary" typeFilterLevel="Full" />
               </serverProviders>
            </channel>
         </channels>
      </application>
   </system.runtime.remoting>
</configuration>.
```

Most of the code for the ServerProcess class is contained in the methods implemented from the ITalkServer interface. The basic outline is shown here:

```
Public Class ServerProcess
    Inherits MarshalByRefObject
    Implements ITalkServer

    ' Tracks all the user aliases, and the "network pointer" needed
    ' to communicate with them.
    Private ActiveUsers As New Hashtable()

    '(Code omitted.)

    Public Function GetUsers() As System.Collections.ICollection
Implements TalkComponent.ITalkServer.GetUsers
    '(Code omitted.)
    End Function

    Public Function AddUser(ByVal [alias] As String, ByVal client As
ITalkClient) As Boolean Implements TalkComponent.ITalkServer.AddUser
    '(Code omitted.)
    End Function

    Public Sub RemoveUser(ByVal [alias] As String, ByVal User As
String) Implements TalkComponent.ITalkServer.RemoveUser
    '(Code omitted.)
    End Sub

    <System.Runtime.Remoting.Messaging.OneWay()> _
    Public Sub SendMessage(ByVal senderAlias As String, ByVal
recipientAlias As String, ByVal message As String) Implements
TalkComponent.ITalkServer.SendMessage
   '(Code omitted.)
    End Sub
End Class
```

6.5 Tracking Clients

The Student Talk .NET server tracks clients using a Hashtable provides several benefits compared to arrays or other types of collections:

- The Hashtable is a key/value collection (unlike some collections, which do not require keys). This allows you to associate two pieces of information: the user name and a network reference to the client.
- The Hashtable is optimized for quick key-based lookup. This is ideal, because users send messages based on the user's name. The server can speedily retrieve the client's location information.
- The Hashtable allows easy synchronization for thread-safe programming.

The collection stores ITalkClient references, indexed by user name. Technically, the ITalkClient reference really represents an instance of the System.Runtime.Remoting.ObjRef class. This class is a kind of network pointer – it contains all the information needed to generate a proxy object to communicate with the client, including the client channel, the object type, and the computer name. This ObjRef can be passed around the network, thus allowing any other user to locate and communicate with the client.

6.6 Sending Messages

The process of sending a message requires slightly more work. The server performs most of the heavy lifting in the SendMessage() method, which looks up the appropriate client and invokes its ReceiveMessage() method to deliver the message. If the recipient cannot be found (probably because the client has recently disconnected from the network), an error message is sent to the message sender by invoking its ReceiveMessage() method. If neither client can be found, the problem is harmlessly ignored.

6.7 File Transfer

The file transfer operation can be broken down into four steps:

1. Peer A offers a file to Peer B.
2. Peer B accepts the file offer and initiates the transfer.
3. Peer A sends the file to Peer B.
4. Peer B saves the file locally in a selected location.

6.8 The Talk Client

The client portion of the Student Talk .NET is called TalkClient. It is designed as a Windows application (much like Microsoft's Windows Messenger). It has exactly two responsibilities: to allow the user to send a message or offer a file to any other online user and to display a log of sent and received messages.

When the TalkClient application first loads, it executes a startup procedure, which presents a login form and requests the IP address of the server and the name of the user that it should register. If one is not provided, the application terminates. Otherwise, it continues by taking two steps:

1. It creates an instance of the ClientProcess class and supplies the user name. The ClientProcess class mediates all communication between the remote server and the client user interface.
2. It creates and shows the main chat form, named Talk, around which most of the application revolves.

On startup, the ClientProcess object registers the user with the coordination server. Because ClientProcess is a remotable type, it will remain accessible to the server for callbacks throughout the lifetime of the application. These callbacks will, in turn, be raised to the user interface through local events.

The login form is quite straightforward. It exposes a public UserName property and a public IPServer property, which allows the Startup routine to retrieve the user name without violating encapsulation.

The ClientProcess class does double duty. It allows the TalkClient to interact with the TalkServer to register and unregister the user or send a message destined for another user. The ClientProcess also receives callbacks from the TalkServer and

forwards these to the TalkClient through an event. The TalkServer will call the ClientProcess is to deliver a message sent from another user. At this point, the ClientProcess will forward the message along to the user interface by raising an event. Because the server needs to be able to call ClientProcess.ReceivedMessage() across the network, the ClientProcess class must inherit from MarshalByRefObject. ClientProcess also implements ITalkClient.

Following is the client configuration, which only specified channel information. The client port is not specified and will be chosen dynamically from the available ports at the runtime. As with the server configuration file, which only specified channel information. The client port is not specified and will be chosen dynamically from the available ports at runtime. As with the server configuration file, you must enable full serialization if you are running the Talk .NET system with .NET 1.1. Otherwise, the TalkClient will not be allowed to transmit the ITalkClient reference over the network to the server.

```
<configuration>
   <system.runtime.remoting>
      <application>
         <channels>
            <!-- The "0" port is declared to allow remoting to
choose -->
            <!-- the most appropriate channel. -->
            <channel port="0" ref="tcp">
               <!-- If you are using .NET 1.1, uncomment the lines
below. -->
               <serverProviders>
                  <formatter ref="binary" typeFilterLevel="Full" />
               </serverProviders>
            </channel>
         </channels>
      </application>
   </system.runtime.remoting>
</configuration>
```

6.9 Chapter Summary

In this chapter, I developed a P2P application using Remoting and showed how it could be modified into a peer-to-peer system with a central coordination server. This chapter also dove into threading intricacies and the heavy lifting need to manage concurrent access with the coordination server.

Chapter 7 System Testing

7.1 Introduction

System errors and failures occur mainly because of inadequate or improper testing. Quality system however demands that system be tested properly. The purpose of testing is to detect the presence of errors in system; errors that have not been discovered yet undiscovered yet. That means, a good test case is one that has a high probability of finding as a yet undiscovered error. A successful test is one that discovers errors where as an unsuccessful test is one that discovers no errors. The goal is to design tests that will uncover the greatest number of errors or classes of errors in minimum amount if time and effort. Successful testing will result in quality system; system with fewer errors and which work according to specification and performance requirements. It will lead to dependable and reliable system.

7.2 Testability

The system designed must be amendable to testing. That is means the system must have the following characteristics.

- Simplicity

The simpler the system, the easier it is to test. System simplicity means that it is functionally and structurally simple and that it follows proper standards.

- Understandability

A system is understandable if its design and interfaces between modules are easy to understand. Understandability also implies that the associated documents are accurate, complete, accessible, and easy to follow.

- Operability

The system must work properly if you want to test it. That is means it must have a fewer errors as possible and that no bugs block the execution of tests.

- Observable

This means the system states and variable are visible or can be queried during execution, all factors affecting the output are visible, source code is accessible, internal errors are detected and reported, distinct outputs are generated for each input, and incorrect outputs are easily identified.

- Controllability

This implies that all codes are executable through some combination of inputs, all possible outputs can be generated through some combination of inputs, inputs/output formats are consistent and structured, and tests can be conveniently specified, automated and reproduced.

- Decomposability

The system is built from independent modules and that each module can be tested independently. This helps to isolate errors and perform re-testing more easily.

- Stability

The system changes are infrequent and controlled; changes do not invalidate existing tests, and recovers well form errors.

7.3 Attributes Of A Good Test

- It should not be redundant, have the same purpose as another test. It will simply waste time and resources. That is means every test must have different purpose.
- It must have a high probability of detecting an error. That is means the tester must picture how the system might fail. Ideally the potential errors are identified and tests are designed to detect those errors.
- It must uncover a whole class of errors.
- Each test case should be executed separately. Combining several tests may be cheaper but it may result in side effects that may mask errors.

7.4 Designing Test Cases

One of the objectives in test case design is to uncover the most number of errors with the minimum amount of time and effort. The test case must have the highest likelihood of detecting errors in the system. A number of approaches or strategies have been proposed to this end with verifying degree of success. They are:

a. Functional Testing

This is intended to exercise or test the functions specified in the system. It derives its test cases from the program specification. Know each of the function must do and design test cases to demonstrate that it works properly.

b. Structural Testing

This is used to exercise or test the internal structure of the system. It derives its test cases from the knowledge of the program's internal structure.

c. Interface Testing

This is to test the user interface. It derives its test cases from the program specification as well as from the knowledge of its interval interfaces. This is especially important in the Windows environment.

The coding and integration stage begins after the detailed design phase is complete. During this phase of the development cycle, modules are coded and documented using the detailed design as blueprint. As each module in Student Talk .NET is written, it is tested for errors and any errors discovered are removed. The modules are then assembled together to build the system. As the modules are integrated, the system is tested. After the integration is completed, the entire system is tested further for errors. It is important to consider ahead of time in order in which

the modules are to be coded and the strategy used to build the system. The approach used in coding the modules and assembling the system is called an integration strategy. There are several alternative methods, each of which has its pros and cons.

Coding and testing are carried out in parallel. The approach chosen to guide integration affects both the progression of the coding and the scheduling of testing activities. The levels of testing include:

- Module Testing

  Tests if the individual modules meet the required specifications and are correctly coded.

- Integration Testing

  Tests if all the modules (when integrated) work correctly. This ensures that the modules are correctly interfaced.

- Function Testing

  Tests if all the functions required by the application and specified in requirements specification document are working properly.

- Performance Testing

  Tests if the performance of the system meets the required specifications. (Non-functional requirements)

- Acceptance Testing

  Tests if the system can be accepted for operation.

- Installation Testing

  Tests if the system works correctly in real environment. This is not necessary if the system is developed in the user's site.

7.5 Testing Strategies

7.5.1 Integration Testing

After performing the unit test, the modules are integrated or combined into a working system. This integration is planned and co-coordinated so that when a failure occurs, it can be solved immediately. There are two major approaches to integration:

- Incremental approach

The modules are added to one by one to the set of already integrated modules, when performing the integration test, the system is viewed as a hierarchy of modules.

- Big-Bang / Non-incremental approach

All modules are combined together in one step.

There are four types of integration testing: Bottom-up, Top-down, Sandwich and Big-Bang testing.

7.5.2 Testing The Entire System

This refers to testing of group of modules, up to and including the entire system. The first level of the testing in the large is the integration testing. Integration testing begins by testing small group of modules and ends with the testing the entire system. Integration testing has two broad purposes:

- To test the interfacing and integration of the modules in the system
- To test the functional performance of the system

Acceptance testing is the second level of testing in the large. In acceptance testing, the user tests the entire system. The goal of this testing is to make sure the system meets the user's requirements.

Once the system tests have been satisfactory completed, the system is ready for acceptance testing, which is testing the system in the environment where it will eventually be used. The purpose of acceptance testing is for users to determine whether the system meets their requirements. The most complete acceptance testing will include alpha testing, where simulated but typical data are used for system testing; beta testing, in which live data are used in the user's real working environments.

During alpha testing, the entire system is implemented in a test environment to discover whether or not the system is overtly destructive to itself or to the rest of the environment. The types of tests performed during alpha testing include the following:

- Recovery testing

  Force the system to fail in order to verify that recovery is properly performed.

- Security testing

  Verifies that protection mechanisms built into the system will protect it from improper penetration.

- Stress testing

  Tries to break the system (for example, what happens when a record is written to the database with incomplete information)

- Performance testing

  Determines how the system performs on the range of possible environments in which it may be used (for example, different hardware configurations,

networks, operating systems); often the goal is to have the system perform with similar response time and other performance measures in each environment.

In beta testing, subsets of the intended users run the system in their own environments using their own data. The intent of the beta testing is to determine whether the software, documentation, technical support and training activities work as intended. In essence, beta testing can be viewed as a rehearsal of the installation phase. Problems uncovered in alpha and beta testing in any of these areas must be corrected before users can accept the system.

The testing process should be precisely specified and set out in the project plan. It is desirable to start test planning at a relatively early stage in the system development process. It is a good practice to start to develop the system and acceptance test suite during the latter stages of the requirements specification. The reasons for preparing the system tests and acceptance tests early are that:

- It helps the developers/analysis in carrying out requirements analysis
- It helps to predict the resources required for system and acceptance testing
- It helps to anticipate special purpose testing tools needed

7.6 Validation And Verification

Validation and verification is the generic term used for checking process, which ensures that the software meets its requirements and that the requirement meets the needs of the user. This process is a whole life cycle process. It starts with requirements review and continues through regular and structured reviews to produce testing main objectives of validation and verification are:

- Discovery of defects in the system
- Assessment of whether or not the system is usable in an operational situation

Validation and verification can be divided into two broad groups, which are dynamic and static. Testing is processing program code to carry out validation and verification. Testing is the main dynamic approach to validation and verification. Prototyping can be viewed as a dynamic requirements validation technique. Examples of static techniques include reviews and formal verification.

7.6.1 Static Verification

This does not involve program execution. Instead it concentrates on the examination of its design. The main objective is to detect errors or to prove that the program is consistent with its specification. However, this type of verification cannot completely replace testing, as it cannot predict the dynamic behavior of programs. There are two broad classes of static verification techniques, which are Review and Formal Verification. Reviews are organized manual approaches to checking and analyzing software (code, design specification, requirement's specification). Typical review processes are planning, overview, preparation, rework,

and follow-up. For formal verification is the most complete analysis technique. When using this, we try to prove that a program meets its specification. Typically, the specification is given in terms of two assertions – preconditions and post conditions, which hold true before and after the system's execution. The system if correct if we can prove that it transforms the preconditions into the post conditions and that it terminates.

7.7 Chapter Summary

In this chapter, we explain about the testing technical that need to use to test the P2P application. During the testing process, all the errors are being corrected.

Chapter 8 System Evaluation

8.1 Introduction

System evaluation is a process of evaluating the developed system to identify the system's strengths and limitations as well as for future enhancements. It also enables the developer to solve the problem encountered during the system's development.

8.2 Problem Encountered And Their Solutions

In order to develop an interesting and user friendly P2P application, I have faced various problems from making the proposal paper project until the development of the system. The following are some of the major problems and approaches taken to solve them from the beginning through the end of the development process.

8.2.1 Inexperienced In Using Programming Language

Visual Basic .NET is a new programming language that actually provides the features that are most important to programmers, such as object oriented programming, strings, graphics, graphical user interface (GUI) components, exception handling, multithreading, multimedia, file processing, database processing, and many others features for implementing Internet based and World Wide Web based applications that seamlessly integrate with windows based applications. However, all these interesting features seem as hopeless and helpless as I never being taught about the language.

Solution:

However, I must make effort on how to explore the language and learned how to use it else I might not be able to complete my system at all. To overcome this problem, much time was really spent in learning and gasping the new language. I learnt from books and tutorials from the Internet. Finally, I am able to learn and understand the language even though there were still rooms for mastering it.

8.2.2 Lack Of Time

I took the WXES 3182 in the final semester and that, I still have many courses that I take along the project and most of them are the third year's papers and most of the paper are hard to score and I need to concentrate more on the courses as well as the project. Besides that, I need to face various assignments, tests and exams where I need to settle down the courses as well as the project. The project itself has taken a lot of my time where it must be developed everyday and that I sometimes felt very stress and hopeless when I come to the hardest part.

Solution:

Here, an effective and maximum time management and scheduling are very important to ensure that every assignment given and the project can be done and settle down in a timely manner, means on time. I have managed my time properly and that; I am able to finish my assignments and project on time.

8.3 System Strengths

After finishing developing my system, I found that there are several strengths in my system that might ease the users to use it. Even though the strengths are not good enough, I am satisfied with it as I said before that I never have experienced in developing such system using the tools that never been used before, and once I am able to make it, I am happy with it. Below are the strengths of my system.

8.3.1 Friendly Graphical User Interface

The major advantage that I figured out from my system is that it provides friendly and easy to use user interfaces. I designed provides friendly and easy to use user interfaces. I designed everything using Visual Basic .NET and I am managed to design interfaces that the system must has. The GUI components such as the server and the client and many more that a P2P application must have. All these GUIs are essential for the users so that they are able to use it.

8.3.2 System Transparency

This refers to the condition where the users do not need to worry about how the system is conducted and structured.

8.4 Current Enhancement

I managed to keep track with what I have proposed in my proposal and I think that there were no big changes except for the User Interface Design. Besides that, this is done to make my system much simpler and yet presentable. I feel more comfortable with the current design and I have received many comments from my friends regarding the user interface and all the comments are very supporting.

Main functions:

- Each client can send messages to each other.
- Each client also can share their files.
- Each client can clear all the received messages.
- Each client can connect/disconnect to a certain central coordination server.

Main Features:

- Friendly user interface using VB .NET coding.
- Mechanisms to handle users who supply duplicate user name.
- Can connect/disconnect to certain central coordinator.
- It allows multiple clients to instant messaging and sharing files.
- Clients-side threading code to properly handle user interface refreshes, particularly when multiple messages are received at once.
- Cleaning up disconnected clients.
- Display time when instant messaging and sharing files.

8.5 Future Enhancements

If there are still times for me to enhance my current system, I would like to make it more interesting and fascinating. There are always new ideas encountered during the development of the system. However, due to time constraints, not all of these ideas could be incorporated into the system. Some of my ideas are:

- Interactive and Dynamic Interface

Additional interactive images such as "GIF" images and Flash can make the system more interesting and attracting.

- Scalability Challenges with the Simple Implementation

In its current form, the Student Talk .NET application is hard pressed to scale in order to serve a large audience. The key problem is the server component, which could become a critical bottleneck as the traffic increases. To reduce this problem, we need to switch to the decentralized approach.

- Firewalls, Ports and Other Issues

Remoting does not provide any way to overcome some of the difficulties that are inherent with networking on the Internet. For example, firewalls, depending on their settings, can prevent communication between the clients and the coordination server.

- Optional Features

Finally, there are a number of optional features that we can add to Student Talk .NET. These include variable user status, user authentication with a password, and buddy lists.

8.6 Chapter Summary

Student Talk .NET presents a straightforward way to reinvent the popular instant messaging and file sharing application in .NET code. However, as it currently stands, it is best suited for small groups of users and heavily reliant on a central coordination server.

Reference

[1] Deitel, Deitel, Nieto. (2002). *Visual Basic .NET.* (2nd Edition). Prentice Hall.

[2] Sommerwille, Ian. (2001). *Software Engineering.* (6th Edition). Addison-Wesley Ltd.

[3] McConnell, Steve. (1996). *Rapid development.* Microsoft Press.

[4] AIM (AOL Instant Messenger)
http://www.aim.com

[5] ICQ
http://www.icq.com

[6] .NET Messenger
http://messenger.msn.com

[7] Yahoo! Messenger
http://messenger.yahoo.com

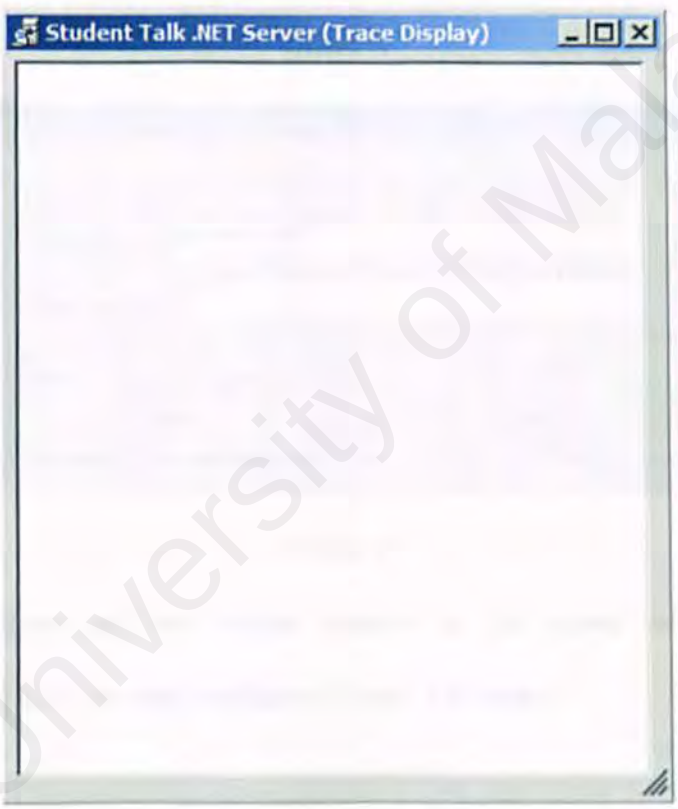
[8] Jerry Lee Ford, Jr. (2000). *Practical Microsoft Windows Peer Networking.* Indianapolis: Que Corporation.

[9] Dana Moore and John Hebeler. (2002). *Peer-to-Peer: Building Secure, Scalable and Manageable Networks.* California: McGraw-Hill/Osborne.

[10] Thomas W. Madron. (1993). *Peer-to-Peer LANs: Network Two to Ten PCs.* Canada: John Wiley & Sons, Inc.

User Manual

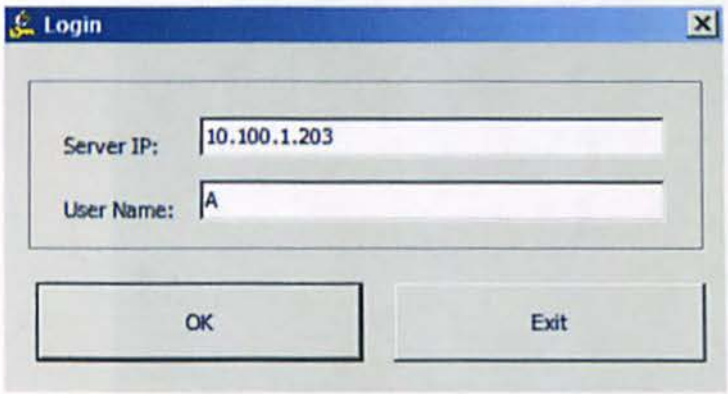
1. Before you can start using the P2P application, Student Talk .NET, you must install Microsoft framework 1.1.
2. You can download the framework 1.1 via Microsoft homepage, it is free and used to support the application being developed using Microsoft Visual Studio .NET.
3. After you have installed the framework 1.1, now you can start running the P2P application, just click on the TalkServer.exe.



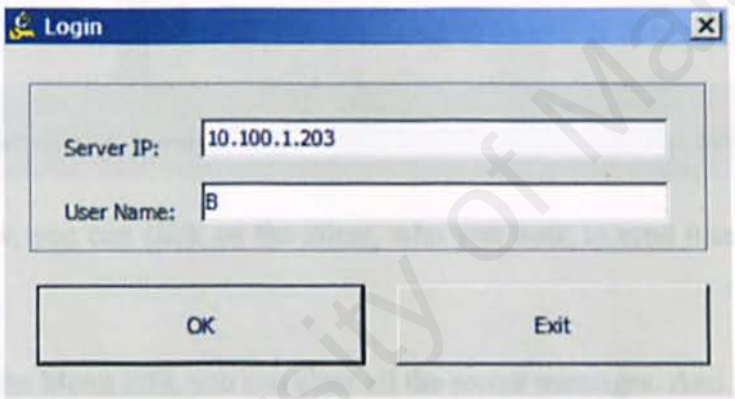
Student Talk .NET Server user interface.

4. You will see the above screen which used to trace all the activities being run in all the client applications, which is connected to it.

5. After you have running the server application, now you just need to click on TalkClient.exe to launch the client application, and type the server IP address which you want to connect to, and the user name.

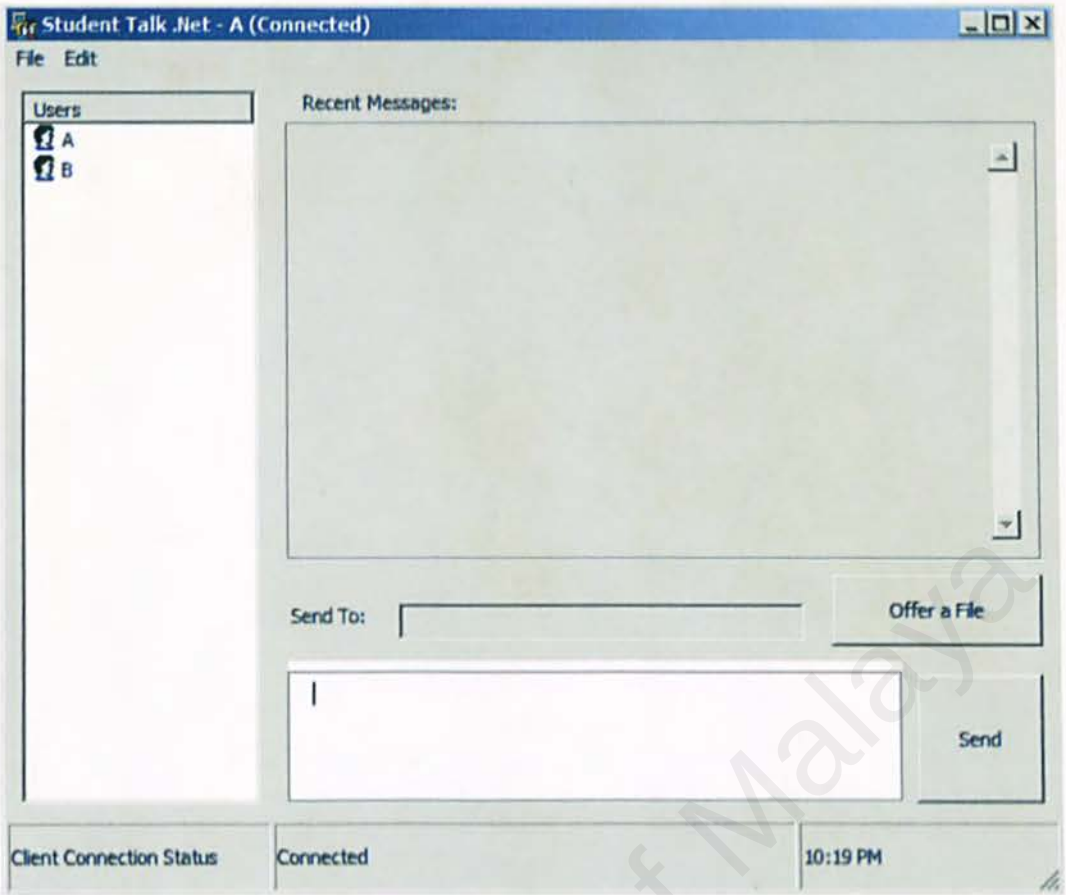


Client A



Client B

6. When there are two clients connect to the server with IP address 10.100.1.203, the user interface of client A as below:



6. Now, you can click on the client, who you want to send message or offer a file.

7 At the Menu Edit, you can clear all the recent messages. And Menu File, you can disconnect and connect to another server or exit the program.